PAGE CUBE: A Model for Storage and Retrieval of Documents Relevant to a Document Production Workflow in an Office

Smriti K. Sinha
Dept. of Computer Science
Tezpur University
Tezpur- 784028, India
email: smriti@agnigarh.tezu.ernet.in

Gautam Barua
Dept. of Computer Science and Engineering
Indian Institute of Technology, Guwahati
Guwahati-781031, India
email: gb@iitg.ernet.in

Abstract

A model for storage and retrieval of relevant documents during a document production workflow in an office is presented. The objective of the workflow is to review a document by multiple reviewers, where each reviewer contributes her own comments during review based on other documents containing standing rules, precedents etc, which is termed, the context. Storage of such documents is modeled as a Page Cube and query languages analogous to relational languages have been proposed for easy and efficient retrieval.

Key Words: *office automation, document production workflow, multi-part multi-signature document, document database, context, multi-dimensional model, page cube, relational query.*

1 Introduction

It is not simple to describe all the work performed in an office in a common framework without referring to the specific organization. But document production and storage is a common work in almost all offices. The scope of our discussion is limited to this common work. Document production in an office is based on a *request-reaction-response* paradigm. When a document containing a request is received in an office, the office reacts to the request. The reactions are recorded in the form of comments on the document and finally a response document is despatched. We can term the process as Document Production Workflow (DPW) [8]. The resultant document of a DPW is termed as a multi-part multi-signature document (MPMSD) [7]. Therefore, a MPMSD is a *case* of the DPW. The first part of a MPMSD is the request document and the last part is the response document and the the parts in between are the comments of the reviewers. Each part of a MPMSD is signed by the corresponding reviewer. The first reviewer is also termed as the originator of the request.

1.1 A Scenario

To understand the salient features of a DPW in an office let us consider the following scenario of a DPW in a University. An employee, A submits an application, m_A regarding her travel plans for approval to the head, B of the department. B verifies the travel plans in the context of previous cases of employees from the department already in travel, type of leave to be granted for A during travel, resolutions on travel taken in departmental advisory committee, standing rules, etc. and adds her comment, m_B and forwards it to the finance officer, C. C also examines the case by verifying the budget allocation status under the head of account for travel, TA/DA rules in such cases, circulars from University Grants Commission on travel expenditure, and adds her comment, m_C on the amount that may be granted and forwards it to the director, D. The director also justifies the previous comments, approves the travel plans and adds the note of approval, may be in the form of office order,m_D. A copy of the whole multi-part document or only the office order m_D may finally go back to the originator, A and the original multi-part document is stored in a folder. The flow of the document is recorded in log-books. This is a case of the travel plan workflow.

1.2 Motivation

In an office, a new document is produced in the context of a set of existing documents constituted of rules, precedents and other support documents. Rules are framed almost on all topics to prevent possibility of arbitrary decisions and are generally well defined. Where rules are not clear, we look for similar cases handled earlier, that is, precedents. Here, rules include regulations, office orders,

0-7695-0996-7/01 $10.00 © 2001 IEEE

meeting proceedings etc. and the precedents are already produced cases following concerned rules and regulations. Certain decisions require support documents (for example, a purchase indent to sanction a purchase). Rules, precedent and such support documents constitute a *reference space*. A user navigates through a subspace of the reference space before producing a new document. This subspace is the *context* of the workflow. The navigation through the context is called the *case examination*. Therefore, the basic objective of the present work is to study automatic retrieval of documents relevant to a DPW and to thus form the context for the case examination. After production, the new document may be linked to different documents of the office. It may, in turn, be included in the context of some other documents. Classical office document architecture proposed by ISO [2] fails to capture the context and the linking of documents.

1.3 The Issues

The problem is basically grouping of office documents in different groups based on different perspectives and to retrieve the relevant documents automatically from the corpus of documents. In this section we enumerate the issues of storage and retrieval of documents in an office with respect to a DPW.

- **Dynamic Context**: The issue is how to construct the dynamic context of a DPW automatically, so that the user can peruse the required documents from the context, while creating a new document. As soon as a new document is created, it is to be incorporated in the concerned contexts automatically.
- **Citation Set**: During creation of a new document, the user can select a subset of documents from the dynamic context and can draw citations at different points of the new document. Thus the document citing and the set of documents cited form a set called the citation set.
- **Static Context**: The context of an already created document is static in nature. It is a snapshot of the dynamic context at the time of creation of the document. Given a document, we can find out the context of the document. It signifies the set of related documents made available to the user at the time of creation of the document.

1.4 Related Work

Document modeling is a well researched problem in Computer Science. There are different approaches for retrieval of relevant documents. In the conventional information retrieval (IR) approach, a document is treated as a long sequence of unstructured text. Searches are performed primarily by matching keywords, usually combined with boolean operators, against the text. Prominent work in this approach are found in [1, 3, 4]. Another approach is mapping document structures and meta-data into a database schema, and importing the documents into database objects of a host system. The capabilities of the host system can then be used to perform queries. The drawbacks of this approach are, conversion process is expensive and the incompatibilities between document structures and database schema often result in loss of information [6]. In another approach documents are represented as graphs with nodes and links. Labels at the nodes or edges in the graph represent the data and the meta-data in the documents, and query processing involves traversal and transformation of the graphs. Prominent works in this approach are [10, 5]. In another approach, documents are considered as fully structured, structured in SGML, HTML or XML and the Document Type Definitions (DTD) are considered as schema and the documents as instances of corresponding DTD and thus documents can be queried using formal query languages. The work by Sengupta [6] is a major work in this direction. Here, the documents of the DocBase are assumed to be structured in SGML. As a result of this, the content of a document is represented as a hierarchy of elements. Formal query languages are designed to retrieve information from such documents.

Normally, office documents have a rich set of attributes like type, topic, time of creation etc. The values of these attributes form the *meta-data* of the document. Our approach deals with a macro-level granularity, where documents are retrieved based on the meta-data of the documents. The documents may be structured, unstructured or semi-structured. Moreover, the documents are highly interlinked. The meta-data is modeled as dimensional attributes and interconnectivity of pages as graphs. An overall architecture of DPW is available in [7] and a secure production protocol for production of MPMSD using a central arbiter is discussed in [8]. Some of the techniques discussed in [6] have been extended to this setting.

2 The Page Cube Model

Here we discuss a model for storage and retrieval of documents in an office during document production workflow with the context as the main binding element. The office documents are considered here as *pages*. We term the model as *Page Cube (PC)*. A PC is a collection of registered pages of an office. Here pages are the main entities. A page has a *profile*, which describes the page and is defined by a set of attribute-value pairs. *Registration* of a page means adding and recording a new page to the page cube and assign a unique pageId to the new page. PC has two components:

page space and *page graphs*.

2.1 The Page Space

The page space is an n-dimensional space defined by n orthogonal dimensions. Each dimension represents a *theme* and is defined by an attribute. An attribute may have attributes and further attributes. Thus, attributes of a dimension form a dimensional hierarchy. Therefore, we can say that the page space is defined by n orthogonal hierarchies. A page is represented in this space as a point, whose coordinate is an n-tuple. The main dimensions include the following but are not limited to:

- **Time**: It is a hierarchical dimension of constant height. The hierarchy is *year.month.day.hour.min.sec*. The time dimension provides the time of creation of a page.
- **Topic**: A page may be in one or more topics. A topic may have subtopics and a subtopic may be further classified. Thus it forms a topic hierarchy. Topic is a growing hierarchy.
- **Type**: A page may be of a type. *office order, circular, notice, comment* etc. Types are basically templates or forms using which pages are generated. We assume in our model that a page is created using some predefined form. A type may have subtypes and thus forms a growing type hierarchy. In time, a new type may be created under an existing leaf type.
- **Category**: A page may belong to one of the three categories: *context, document* or *part*. The pages of category part are the elementary pages. A page of category document contains a set of links to the pages of category part. The links are ordered on the time of creation of the parts. The pages of category documents are the MPMSDs. A page of category context contains a set of links to pages of category document. A DPW will have a context page associated with it.
- **Class**: Pages may be classified based on the themes of the content of the pages. The classes are *rule, case* and *support*. A page may contain rules on some topics, may be a part of a case of a workflow or may be a support page. A page may belong to more than one class. Content of a part of a case may be rules on some topics. For example, resolutions of the Board of Management (BoM) is the last part of a case of BoM meeting workflow, at the same time, it is a rule by default to concerned topics. Since page cube is a closed model, as discussed later, where a result-set of a query will also be a page containing links to the pages satisfying the query and thus serves as a hub, therefore *resultset* also may be a class. Moreover, the queries are also stored in the page cube as pages, therefore *query* may be another class.
- **User**: In DPW, user is an important dimension. A user belonging to an office may be the reviewer of some cases and therefore the author of some pages. For some cases, like the pages of category document or context, the author is the arbiter itself.
- **Domain**:The users of an office may belong to different domains of the office. Therefore, a page may be originated from a domain of an office. An office may have many domains and subdomains.
- **DPW**: In our model a page is produced in a DPW. This dimension gives the concerned workflow of a page.
- **State**: The pages in a DPW will be in one of the three states: *active, reference* or *archive*. Active pages are the pages, whose content are changing. When an active page is closed it becomes a reference page. The old pages which are neither active nor are reference are in archive state. The characteristic which differentiates these states is the accessibility. Archive pages have no access privileges for the users. The reference pages are read only and active pages may have privileges like read, write, modify etc.

This set of dimensions, common for all DPWs, is only a representative one. An office can identify more useful dimensions specific to the office concerned.

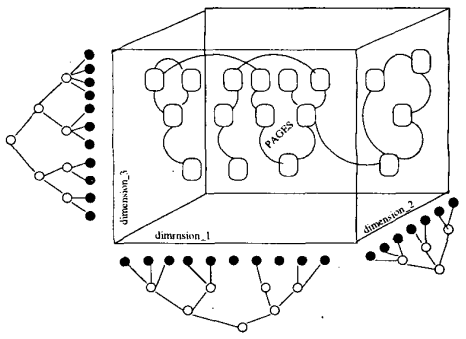

Figure 1. Page Cube

2.2 The Page Graph Component

The pages of a PC are linked to a given page either implicitly or explicitly. Implicitly linked pages are those pages which satisfy a *predicate* defined over the dimensional values of the pages. Explicitly linked pages are those pages which are linked by explicit hyperlinks. Thus, the pages, which are explicitly linked, form a directed graph, where

the pages are the vertices and the hyperlinks are the edges. In addition to the implicit links provided by the attributes of the dimensional hierarchy, the pages belonging to a dimension may be explicitly linked forming a *dimensional graph(DG)* of the concerned dimension. Thus the page graph component of PC is a set of DGs.

Let V be a set of pages belonging to a dimension d, E a set of directed edges among these pages. Then, $G_d(V, E)$ is a digraph representing a dimension graph of dimension d. A link in PC is a bidirectional link and is represented in $G_d(V, E)$ as a conjugate pair of edges, one forward and one backward. The vertices and the edges of $G_d(V, E)$ can be further classified into different kinds, based on the values of the dimensional attribute. Therefore, V and E can be defined as :
$V = \{V_\alpha | V_\alpha$ is a set of vertices of kind $\alpha\}$ and
$E = \{E_\beta^+, E_\beta^- | E_\beta^+$ is a set of forward edges of kind β and E_β^- is a set of reverse edges of kind $\beta.\}$.
Moreover, the edges in E may be weighted, making $G_d(V, E)$ a *weighted graph*. The weight may be defined differently in different DGs. Linking of different kinds of pages by different kinds of edges in $G_d(V, E)$ is subject to the satisfaction of a set of constraints C_d. The act of linking two pages by creating a pair of conjugate edges is termed as *plugging*.

A page may migrate from one storage to the other as soon as it changes its state and accordingly the physical address of the page may change from time to time. But at a particular time a page will have one and only one physical address. Therefore, each page in the page cube will have a pair of addresses: a time varying physical address and an invariant logical address. Let *pageId* be the unique invariant logical address of a page. A page of a particular type may be on more than one topic. In that case, a page may have multiple n-tuple coordinates. That is the reason, a distinct pageId is chosen as the logical address of a document. Otherwise the n-tuple coordinate would have been an ideal logical address of a page. Without any loss of generality, we can use the values of Time dimension of a page as the unique pageId, since in our scheme the timestamp will be given by a central arbiter [8].

3 The Category Graph

Among the different common dimensions discussed in section 2.1, the dimension *category* is a very important dimension. For a DPW, the construction of dimensional graph for category is a mandatory one. Therefore, we shall discuss the category graph in detail in this section. A page may belong to one of the three categories: *part, document* and *context*, as discussed earlier. A part may cite other parts during production of the part. Similarly a part may be cited in many parts. Therefore, pages of category part are linked by *cite* kind of edges. Citing is a special plugging. A document may plug many parts and a part may be plugged in many documents. Therefore, pages of category document and part are linked by *docPart* kind of edges. A context may plug many documents and a document may be plugged in many contexts. Therefore, pages of category context and document are linked by *conDoc* kind of edges. Of course, the model has no restriction on document and document or context and context linking by corresponding kinds of edges but we leave it to the design and implementation of office specific systems. For brevity, we will restrain from discussing about that here.
Let P be a page cube, $T_V = \{part, document, context\}$ is a set of kinds of vertices, $T_E = \{cite, docPart, conDoc\}$ is a set of kinds of edges,
$V = \{V_{part}, V_{document}, V_{context}\}$, where V_{part} is a set of vertices of kind *part*, $V_{document}$ is a set of vertices of kind *document* and $V_{context}$ is a set of vertices of kind *context*. $E = \{E_{cite}, E_{docPart}, E_{conDoc}\}$, where E_{cite} is a set of edges of kind *cite*, $E_{docPart}$ is a set of edges of kind *docPart* and E_{conDoc} is a set of edges of kind *conDoc*. Each E_T, $T \in T_E$, in turn consists of two sets: a set of forward links of kind T, E_T^+ and a set of reverse links of kind T, E_T^-.
The category graph is defined as $G_{category}(V, E)$. The edges in E are weighted. A page may be plugged to another page at some time and may be unplugged at some other time. The weight of an edge is a 2-tuple (t_+, t_-), where t_+ is the time of plugging and t_- is the time of unplugging. The significance of the temporal weight is that an edge remains active till it is unplugged, that is, during the period defined by t_+ and t_-. Therefore, edges in $G_{category}(V, E)$ are persistent in nature. Unplugging does not delete the conjugate pair of edges, only modify their weights. Therefore, the temporal weights keep the history of plugging pages. This temporal weight is used in reconstruction of the context of a part and will be discussed later. Since there is a possibility of plugging as well as unplugging more than once between the same pair of vertices, parallel edges with different weights may exist.

The graph $G_{category}(V, E)$ is constructed by plugging the pages subject to the satisfaction of the following constraints:

- **Citation Constraints**: A page p_i cites another page p_j iff the following conformability conditions on operands are satisfied: (i) $p_i, p_j \in V_{part}$ (ii) $p_i \notin P$ and $p_j \in P$, means at the time of citing, citing page p_i is a new page and the cited page p_j is from P. Only a page belonging to P can be cited in a new page.

- **DocumentPart Constraints**: A page p_j is plugged to page p_i iff the following conformability conditions are satisfied: (i) $p_i \in V_{document}$ and $p_j \in V_{part}$ (ii) p_i

is in active state. (iii) $p_i, p_j \in P$ means, at the time of plugging the operands should be in the page cube. The significance of this restriction is that generation of pages of categories document and context is an internal process of the page cube and confined to registered pages only. For example, when a new page of category document is to be created, first a null document is created then other pages are plugged.

- **ContextDocument Constraints**: A page p_j is plugged to page p_i iff the following conformability conditions are satisfied: (i) $p_i \in V_{context}$ and $p_j \in V_{document}$ (ii) p_i is in active state. (iii) $p_i, p_j \in P$

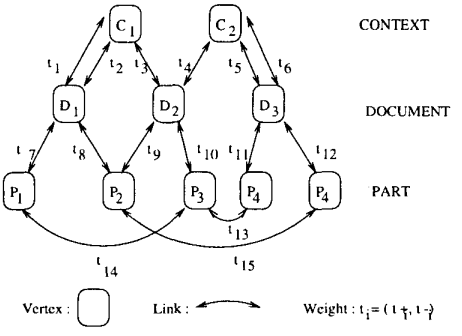

Figure 2. Category Graph

3.1 Closure

Page Cube is a closed model. Simply put, this means that input to a query are pages belonging to PC and the resultset, that is, the output of a query, is also a page having hyperlinks to the pages satisfying the query. Closure is prominent in a relational database model, as tables are both input to queries and output from queries. In addition using the QBE query language, queries are formulated also using tables. Similarly, in page cube, queries can also be pages. The advantages of a closed model are discussed in [6]

4 Production of Contexts

In this section, we discuss the production of dynamic context of a workflow as well as the static context of a part of a MPMSD.

4.1 Production of Dynamic Context

We recall, a context is a collection pages of category document, relevant to a dpw. A DPW has only one context page. Relevance of pages to a DPW is defined by the designer of a dpw using a set of *requirement templates*. A reviewer other than the designer of the DPW cannot delete or modify the given default context but can expand the context by adding new templates. A requirement template is defined by a set of attribute value pairs. The attributes are dimensional attributes of the pagecube. For example, T_1 = {category="document", class="rule", topic="s1", time="t1-t2"} signifies that documents containing rules on $s1$ produced during the tenure $t1 - t2$ are relevant to the DPW concerned.

The production rules for contexts are:

C1. when a new requirement template is submitted for inclusion in the template list, a list of pages from the cube, satisfying the template, are generated and the pages are plugged to the context if not already plugged.

C2. when a new page of category = "document" is registered and if the page satisfy any one of the requirement templates of the dpw, then the page is plugged to the concerned context page.

C3. when a template is submitted for exclusion from the requirement template list, a list of pages, satisfying the template, is generated and the pages are unplugged from the context iff the document do not satisfy any other template belonging to the list.

C4. when a page of category ="document" is de-registered and if the page is plugged to one or more context pages then the document is unplugged from the concerned pages.

4.2 Production of Static Context

Production of the static context of a given page of category part is the process of recreation of history. The basic objective is to regenerate the context prevailing at the time of production of a given page. The static context of a given page is the set of documents plugged to the context at the time of production of the given page. Some of these documents may have been subsequently unplugged, and so the context has to be recreated. From the page cube we can regenerate the static context of a given page using the following algorithm.

Algorithm to regenerate the static context of a given page

Let $p \in P$ be a given page of category part belonging to the page cube P and t_p be the time of production of p. We are to regenerate the context C_{t_p} of p.

BEGIN

1 Find the dpwId w of the part p from P

2 Find the pageId p_c of the page of category = "context" and dpw = "w" from P

3 Find the set of forward edges, e_c^+, from $G_{category}(V,E)$, where $e_c^+ = \{e = (v_i, v_j, (t_+, t_-))|e \in E_{conDoc}^+$ and $v_i \in V_{context}$ is the initial vertex , $v_j \in V_{document}$ is the terminal vertex of the edge and $v_i = p_c$ and $t_+ < t_p$ and ($t_- = 0 \vee t_- > t_p$)}.

4 Produce the context C_{t_p} using e_c^+.

5 On selection of a document with pageId $p_d \in C_{t_p}$

do

5.1 Find the set of edges from $G_{category}(V,E)$, $e_d^+ = \{e = (v_i, v_j, (t_+, t_-))|e \in E_{docPart}$ and $v_i \in V_{document}$ is the initial vertex , $v_j \in V_{part}$ is the terminal vertex of the edge and $v_i = p_d$ and $t_+ < t_p$ and ($t_- = 0 \vee t_- > t_p$})

5.2 Produce a document D_{t_p} using e_d^+.

enddo

END

Here, the pages C_{t_p} and D_{t_p} are not actual pages but the states of the concerned pages at time t_p.

5 Query Languages

A query is a an expression denoting a set of pages described by a formula ϕ of the form $\{p|\phi(p)\}$. A query language provides a user with a means of expressing questions in the form that can be handled by the model enabling the model to answer the questions asked in a reasonable time. In this section we describe two equivalent query languages: The first language is Page Algebra (PA), a procedural language which uses specialized operators on the sets of pages to specify queries and a Page Structured Query Language (PSQL), a user-friendly pseudo-natural language with a simple means for expressing queries using a natural language form. The languages are similar to Relational Algebra and SQL respectively.

5.1 Pages

Pages are the main entities to deal with. Therefore, before we discuss the query languages, we should have a clear understanding of a page and its constituent elements in the context of the page space and page graph components of PC. To define a page, we first define a few sets as follows: Let S be a countably infinite set of strings, $A \subset S$ be a set of variables called attributes, $V \subset S$ be a set of allowed values for the attributes in A, L a set of edges defined as $L = \{(p_i, p_j, w, t, dir)|p_i$ is the source page, p_j is the target page, w be the weight, t be the kind of the edge and dir be the direction of the edge: either forward(+) or the backward(-)}. Source page p_i is always the page containing the edge. Therefore, it is implied and need not be explicitly mentioned.

Definition 1 *A profile B is defined as a 2-tuple, an attribute and a list of values,* $B = \{(a, \{v_1, v_2, \cdots, v_n\})|a \in A, v_i \in V, i = 1, 2, 3, \cdots, n\}$

Definition 2 *A page p is defined as a quadruple* $p = (B, X, L^+, L^-)$, *B is the profile,* $X \subset S$ *is a set of strings defining the content of the page,* $L^+ \subset L$ *is the set of forward edges,* $L^- \subset L$ *is the set of reverse edges.*

5.2 Path Expression

A Path Expression(PE) defines a path from one node in the graph to another in terms of intermediate node and edge labels. PEs in graphs are used in navigation oriented queries. In our model, navigation in the dimensional graphs is a common feature for the queries. Moreover, the dimensions of the PC are also hierarchical. Therefore, values of the attributes can be expressed as PEs. Use of path expressions in document databases are available in [9]. We follow the simplified PE discussed in [6]. The standard wildcat character "*" is used in a path expression to signify all successors of a node in the dimension hierarchy as well as in the dimensional graph. The standard "." operator, commonly used to denote relation attributes can now be cascaded to express a listed path. In addition a ".." operator is introduced, which is used to construct an abbreviated path from a listed path. Details of PE is available in[6].

5.3 Page Algebra

Page Algebra(PA) is an operator based query language for querying pages from a PC. It is an extension of relational algebra. PA is defined in terms of special set operators that map one or more sets of pages to a new set of pages. The result set of an operation in PA is a page conforming to our original notion of closed model. Every PA expression E represents a set of pages. The main operators are as follows:

- **Selection** (σ): The selection operation $\sigma_\gamma E$ extracts a subset of pages from an input set E that satisfies the selection condition γ
  $\sigma_\gamma E = \{p|p \in E \wedge \gamma\}$

- **Plug** ($\times$): Given two PA expressions E_1 and E_2, the expression $E_1 \times E_2$ reproduces the pages belonging to E_1 with forward edges to the pages belonging to E_2 and the pages belonging to E_2 with backward edges to pages belonging to E_1.
  Let $E_1 = \{p_1, p_2\}$ and $E_2 = \{p_3, p_4\}$. $E_1 \times E_2$ produces the pages with the following forward (+) and

backward (-) edges.

$p_1 = (p_1, \{p_3^+, p_4^+\})$ $\quad p_3 = (p_3, \{p_1^-, p_2^-\})$

$p_2 = (p_2, \{p_3^+, p_4^+\})$ $\quad p_4 = (p_4, \{p_1^-, p_2^-\})$

The pages p_1 and p_2 each contains forward edges to pages p_3 and p_4 and similarly the pages p_3 and p_4 each contains backward edges to pages p_1 and p_2 and all the pages are reproduced. Again the edges may be different kinds, say context to document(cd), document to part(dp), part to part(pp). Accordingly the forward and backward edges may be qualified. If E_1 contains pages of category document and E_2 contains parts, then the qualified edges due to $E_1 \times E_2$ will be

$p_1 = (p_1, \{p_3^{dp+}, p_4^{dp+}\})$ $\quad p_3 = (p_3, \{p_1^{dp-}, p_2^{dp-}\})$

$p_2 = (p_2, \{p_3^{dp+}, p_4^{dp+}\})$ $\quad p_3 = (p_3, \{p_1^{dp-}, p_2^{dp-}\})$

- **Unplug** ($\div$): This is the reverse operation of plug. Given two PA expressions E_1 and E_2, the expression $E_1 \div E_2$ reproduces the pages belonging to E_1 deactivating forward edges to the pages belonging to E_2 and also reproduces the pages belonging to E_2 deactivating the backward edges to pages belonging to E_1. Unplug does not necessarily remove the edges physically. It may simply modify the weights of the edges.

- **Path Selection** ($\circ$): Given a PA expression E and a PE P, $E \circ P$ returns the set of pages rooted at $last(P)$ obtained by traversing the path P from each of the pages in E [6].

- **Union** ($\cup$): Union is the normal set union operation. Given two PA expression E_1 and E_2. The result of $E_1 \cup E_2$ is a set of pages defined as
  $E_1 \cup E_2 = \{p | p \in E_1 \vee p \in E_2\}$

- **Intersection** ($\cap$): Intersection is the normal set intersection operation. Given two PA expression E_1 and E_2. The result of $E_1 \cap E_2$ is a set of pages defined as
  $E_1 \cap E_2 = \{p | p \in E_1 \wedge p \in E_2\}$

- **Difference (-)**: is the normal set difference operation. Given two PA expression E_1 and E_2. The result of $E_1 - E_2$ is a set of pages defined as
  $E_1 - E_2 = \{p | p \in E_1 \wedge p \notin E_2\}$

For example, let the query be, find from the page cube P, all the documents containing rules on special casual leave and include them in the context of the dpw "w". In PA it can be expressed as

$\sigma_{category="context" \wedge dpw="w"} P$

$\times$

$\sigma_{category="document" \wedge class="rules" \wedge topic="leave.casual.spl"} P$

5.4 Page Structured Query Language

Here we present a language, called PSQL, for interactively processing queries on pages . PSQL is an extended version of SQL. The primary motivation behind such a language is to provide users of database systems with a simple means for expressing queries using a natural language form. Standard SQL deals with flat tables and the result set of an SQL query is also a table. In SQL the main retrieval operation is the SELECT operation. To accommodate this feature in PSQL the SELECT clause will have the mechanism to allow the creation of composite page from the constituent pages. The resultant page of a query is basically a multi-part page which contains hyperlinks(edges) to the constituent pages.

For example let us take the following query

```
SELECT  * FROM P
WHERE category = "document"
and class = "rules"
and topic IN {"4.3.*', "8..5"}
and time BETWEEN {"1990.08.*", now}
GROUPBY topic, time
```

The query produces a resultant page from a page cube P containing links to all the pages containing rules on any of the topics 4.3 or a sub topic of it, or on a topic 5, such that there exists a path from topic 8 to topic 5, produced between August 1990 and *now*. The value of *now* is defined by the arbiter with the current time stamp. The values included in the IN list are generally values of dimensional attributes. It is a shorter form of expressing ORs of attribute-value based predicates. The range of values in BETWEEN clause is a more general expression of IN list.

The expressive power of PSQL and PA are equivalent. The expressions in PA can be expressed here in terms of clauses. For Example, the query discussed in PA can be expressed as follows:

```
(SELECT * FROM P
 WHERE category = "context" and
       dpw = "w")
 PLUG (SELECT * FROM P
       WHERE
       category = "document" and
       class = "rule" and
       topic = "leave.casual.special" )
```

The complete grammar of PSQL is under development.

6 Relational Structure

The page cube model can be mapped to relational model. In this section we will provide the relational representation of the model.

6.1 The Schema

The profile component can be represented by the star schema or the snowflake schema of multi-dimensional data cube model of data warehousing. It is a common practice in multi-dimensional modeling that when the dimensions are simple in nature, such dimensions are incorporated into the central fact tables as attributes when there is no justification of maintaining a separate dimension table. The advantage is that expensive join operations between the central fact table and those dimension tables can be avoided thereby improving the efficiency. Of course, there is no clear rule for such justification. It is a design decision.

In the schema of the page cube some of the dimensions are incorporated in the central fact tables. Time, Category, Class, Status are such dimensions which can be incorporated to the central tables without any loss of generality. Time is a significant dimension and needs a special treatment for it. Once time is incorporated as an attribute of the central tables then there is no need for a separate timeId. Since in our scheme central arbiter will provide the time, a unique time can be assigned by the arbiter to a page. Therefore, it can also be used as a unique identifier for a page. In that case, we don't need to have a separate pageId. According to this modification, the arbiter ensures that at a particular time one and only one page will be created. The star schema of PC is as follows:

Dimension Tables:
Type (*typeId, typeDesc*, parent.....)
Topic (*topicId, topicDesc*, parent.....)
User (*userId, userName, address*, ...)
Domain (*domainId, domainDesc*, parent .....)
Dpw (*dpwId, dpwDesc*, ...)

Central Fact Tables:
userDpw(*userId, dpwId*, ...)
Production (*pageId, topicId, typeId, category, class, userId, domainId, dpwId*)
Storage(*pageId, location, size, signature, status, authorization*)
Flow(*pageId, senderId, sentTime, receivedTime, ReceiverId*)
Citation(*citingPageId, citedPageId, citeCount*)
DocumentPart(*documentId, partId, plugTime, unplugTime*)
ContextDocument(*contextId, documentId, plugTime, unplugTime*)

Notes: The names of the relations and of the attributes are chosen in such a way that they are self explanatory. For brevity, we did not provide a data dictionary for the schema. A page may be on more than one topic, may also belong to more than one class. Therefore, normalized Production relation will have more than one tuple with same pageId. Therefore, pageId alone cannot be the primary key of the relation. Without any loss of generality, if we consider multiple values of topicId as well as of class as a single atom, where values are delimited somehow, then the given relations serve the purpose. De-normalized relations are allowed in star schema.

7 Conclusion

In the present work we proposed a model, called Page Cube, for storage and retrieval of documents during a document production workflow. The model provides the mechanism to construct the dynamic context of a DPW automatically. It also allows to reconstruct the context of a comment on a precedent. The model can be mapped to the relational model. Correspondingly, PSQL can be mapped to SQL. But detail formalisms of the mapping remains to be future work. The complete grammar for PSQL is under development.

References

[1] A. K. Jain, et.al. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):265–320, 1999.

[2] K. K. Bajaj. *Office Automation*. Macmillan India Ltd., 1 edition, 1989.

[3] C. Zaniolo et.al., editor. *Advanced Database Systems*. Morgan Kaufmann, 1 edition, 1997.

[4] D. Lee et. al. Document Ranking and the Vector-Space Model. *IEEE Software*, March/April, 1997.

[5] S. Abiteboul et.al. The Lorel Query Languages for Semistructured Data. *http://www-db.stanford.edu/ ~ lore/*.

[6] A. Sengupta. *DocBase - A Database Environment for Structured Documents*. Ph. D. Thesis, Indiana University, 1997.

[7] S. K. Sinha and G. Barua. An Architecture for Document Production Workflow in an Office. In *Proc. of the Int. Conf. on Information Technology (CIT99), India*, pages 45–52, 1999.

[8] S. K. Sinha and G. Barua. Secure Flow of Persistent Multi-Part Documents in an Office. In *Proc. of the Int. Conf. on Information Technology at the Dawn of New Millennium, Bangkok, Thiland*, volume 3, pages 157–172, 2000.

[9] V. Christophides et.al. From structured documents to novel query facilities. *SIGMOD RECORD*, 23(2):313–324, 1997.

[10] W. Schuetzelhofer et. al. Graphical Navigation in XML-Databases. In *Proc. of the Int. Conf. on Information Technology at the Dawn of New Millennium, Bangkok, Thiland*, pages 47–59, 2000.