Evaluation of DBMSs Using XBench Benchmark

M. Tamer Özsu and Benjamin B. Yao
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{tozsu,bbyao}@uwaterloo.ca

Technical Report CS-2003-24
August, 2003

Contents

List of Figures

List of Tables

Abstract

XML support is being added to existing database management systems (DBMSs) and native XML systems are being developed both in industry and in academia. The individual performance characteristics of these approaches as well as the relative performance of various systems is an ongoing concern.

XBench is a family of XML benchmarks which recognizes that the XML data that DBMSs manage are quite varied and no one database schema and workload can properly capture this variety. Thus, the members of this benchmark family have been defined for capturing diverse application domains.

In this report we briefly discuss the XBench XML benchmark and report on the relative performance of three commercial DBMSs: X-Hive, DB2 and SQL Server. We also describe the potential issues when storing XML documents into relational DBMSs.

Chapter 1

Introduction

XBench is a family of benchmarks developed to measure and evaluate the performance of XML Database Management Systems (DBMS). The benchmark specification is given in [7] which can also be obtained from `http://db.uwaterloo.ca/~ddbms/ projects/xbench/`. In this document, we report the performance evaluation of three DBMSs using XBench: XHive, which is a native XML system, IBM DB2 and Microsoft SQL Server, which are relational DBMSs with XML support. Although this document summarizes some aspects of XBench (Chapter 2), it does not provide a full description of the benchmark.

Following the overview of XBench in Chapter 2, we discuss the mapping of the benchmark database to relational databases if necessary (Chapter 3). This is followed, in Chapter 4, by a report of the performance of all DBMSs using the XBench workload and the mapping described in Chapter 3. These performance results consist of two parts: the first part of results are obtained without special optimization of the database; the experiments are re-run and the results are obtained again after we created indexes on the tables of the databases.

Chapter 2

XBench Overview

2.1 XBench Database

XBench characterizes database applications along two dimensions[1]: application characteristics, and data characteristics. Application characteristics indicate whether they are data-centric or text-centric. Datacentric (DC) applications deal with data that may not originally be in XML. Examples include e-commerce catalog data or transactional data that is captured as XML. Text-centric (TC) applications manage actual text data natively encoded as XML documents. Examples include dictionaries, book collections in a digital library, or news article archives.

In terms of data characteristics, two classes are identified: single document[2] (SD) and multiple document (MD). The single document case covers databases, such as an e-commerce catalog, that consist of a single document with complex structures (deep nested elements), and dictionaries, while the multiple document case covers those databases that contain a set of XML documents, such as an archive of news documents or transactional data. The result is a requirement for a database generator that can handle four cases: DC/SD, DC/MD, TC/SD, and TC/MD (Table 2.1).

	SD	MD
TC	Online dictionaries	News corpus, Digital libraries
DC	E-commerce catalogs	Transactional data

Table 2.1: Classes of XML databases and example applications

2.1.1 Text-Centric Document Databases

The common features of XML documents in the TC/SD class are a big text-dominated document with repeated similar entries, deep nesting and possible references between entries. The generated XML document is a single big XML document (*dictionary.xml*) with numerous word entries. The size of the database is controlled by a parameter called *entry_num*. The default value of entry_num is 7333 and file size is about 100 MB.

Figure 2.1 gives a visual representation of the schema of XML documents in this class. This diagram is generated using XML Spy, a popular XML editor. It clearly illustrates the parent/child tree structure of element types in a XML documents. The rectangles refer to element types. Whatever is on the right side of an element type are its sub-element types. Solid rectangles mean element types are mandatory while dotted ones mean they may or may not exist in a given document. By default, only one instance of a particular element type can appear in real documents, unless otherwise

[1]The justification for this design can be ignored for this discussion; it is explained in the benchmark specification.
[2]Document, in this context, refers to an XML document.

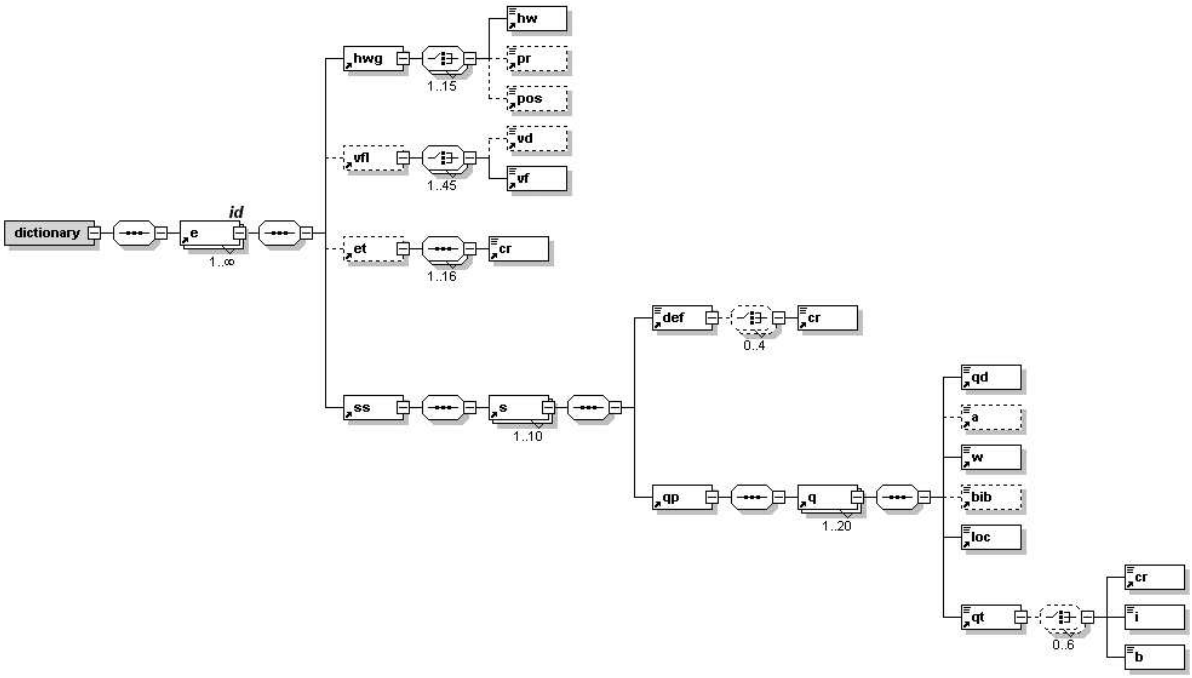

Figure 2.1: Schema Diagram of TC/SD (Dictionary)

specified under rectangles. (0..1) means element types are not mandatory and can appear multiple times, and similarly (1..1) means element types are mandatory and can appear multiple time. If an element has attributes, they will appear on the top right corner of the rectangle where the element is located. The complete XML Schema and DTD files for all database classes are given in [7].

The features of XML documents in the TC/MD class are numerous relatively small text-centric XML documents with references between them, looseness of schema and possibly recursive elements. The target XML documents are a set of XML articles with sizes ranging from several kilobytes to several hundred kilobytes. The size of this database is controlled by *article_num* with a default value of 266, and the default data size is around 100 MB.

Figure 2.2 illustrates the schema information of XML documents in this class. The figure depicts the irregularity of this class of documents.

2.1.2 Data-Centric Document Database

For data-centric classes, the availability of real XML data for analysis is problematic. Although there are XML specification of transactional data (e.g., Electronic Catalog XML (eCX)[3], Commerce XML (cXML)[4], XMLPay[5], and XML Common Business Library (xCBL)[6]), they are not yet widely used and no substantial repository of XML documents can be found. The data that is available is too small to extract meaningful statistics. Most of the XML documents in the data-centric classes are currently relational that may be translated into XML for communication. Therefore, the schema of the TPC-W benchmark is used and is mapped to XML. TPC-W considers a Web-based e-commerce system and can represent DC classes of documents.

There are eight basic individual tables (relations) in the TPC-W database: ORDERS (purchase order information), CC_XACTS (information on credit card transaction for each order), ORDER_LINE (information on all detailed items of each order), CUSTOMER (customer information), ITEM (infor-

[3]http://www.ecx-xml.org
[4]http://www.cxml.org
[5]http://www.verisign.com/developer/xml/xmlpay.html
[6]http://www.xcbl.org

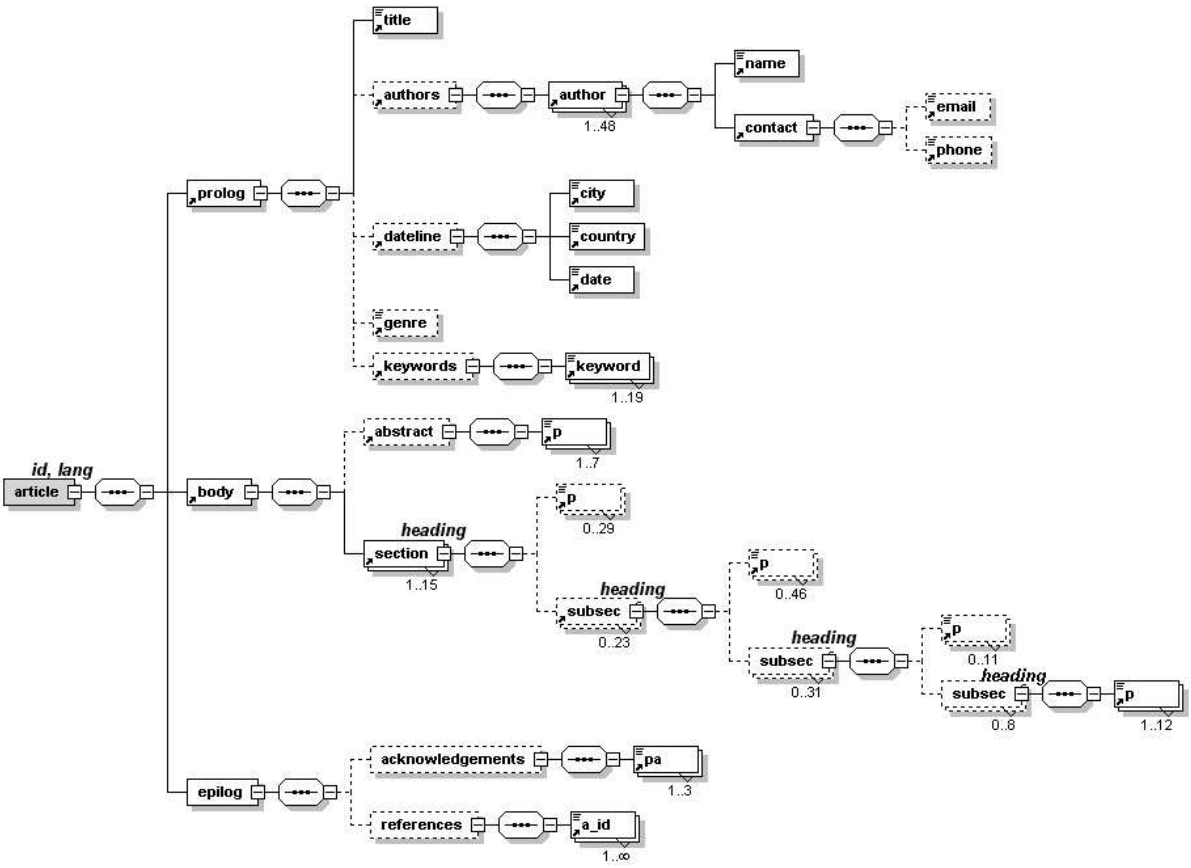

Figure 2.2: Schema Diagram of TC/MD (ArticleXXX)

mation about TPC-W item, which are books), AUTHOR (author information), ADDRESS (address information for each customer), and COUNTRY.

XML documents belonging to the DC/SD class are similar to TC/SD in terms of structure but with less text content. However, the schemas of these data tend to be more strict in the sense that there is less irregularity in DC/SD than in TC/SD, since most of the XML documents in DC/SD are translated directly from relations.

In order to create a catalog data structure from TPC-W relational data model, table ITEM is picked as base, along with the AUTHOR, ADDRESS and COUNTRY tables. Two more tables are created that do not exist in TPC_W: AUTHOR_2 to include additional author information such as: mailing address, phone and email information, and PUBLISHER to record publisher information consisting of name, fax, phone and email address.

These six tables are joined together and mapped to an XML document called *Catalog*. By joining all these tables, more depth is added to *Catalog*. In this mapping, we follow an approach that takes into consideration the domain-specific semantics. The mapping of a join of several tables is done by picking one main table (in our case ITEM) and mapping it to XML first using the above described method. All matching tuples in the second table are inserted as sub-elements of the corresponding tuples in the first table based on foreign key references. The process is repeated recursively for other tables. Accordingly, as the number of joined tables increases, the mapped XML document gets deeper[7]. The visual representation of the tree structure of all element types are shown in Figure 2.3.

[7]There are other mapping techniques that have been proposed such as flat transaction [6, 4], nesting-based translation [4], and constraints-based Translation [5]. Review of the problems in using these for our purposes is given in [7].

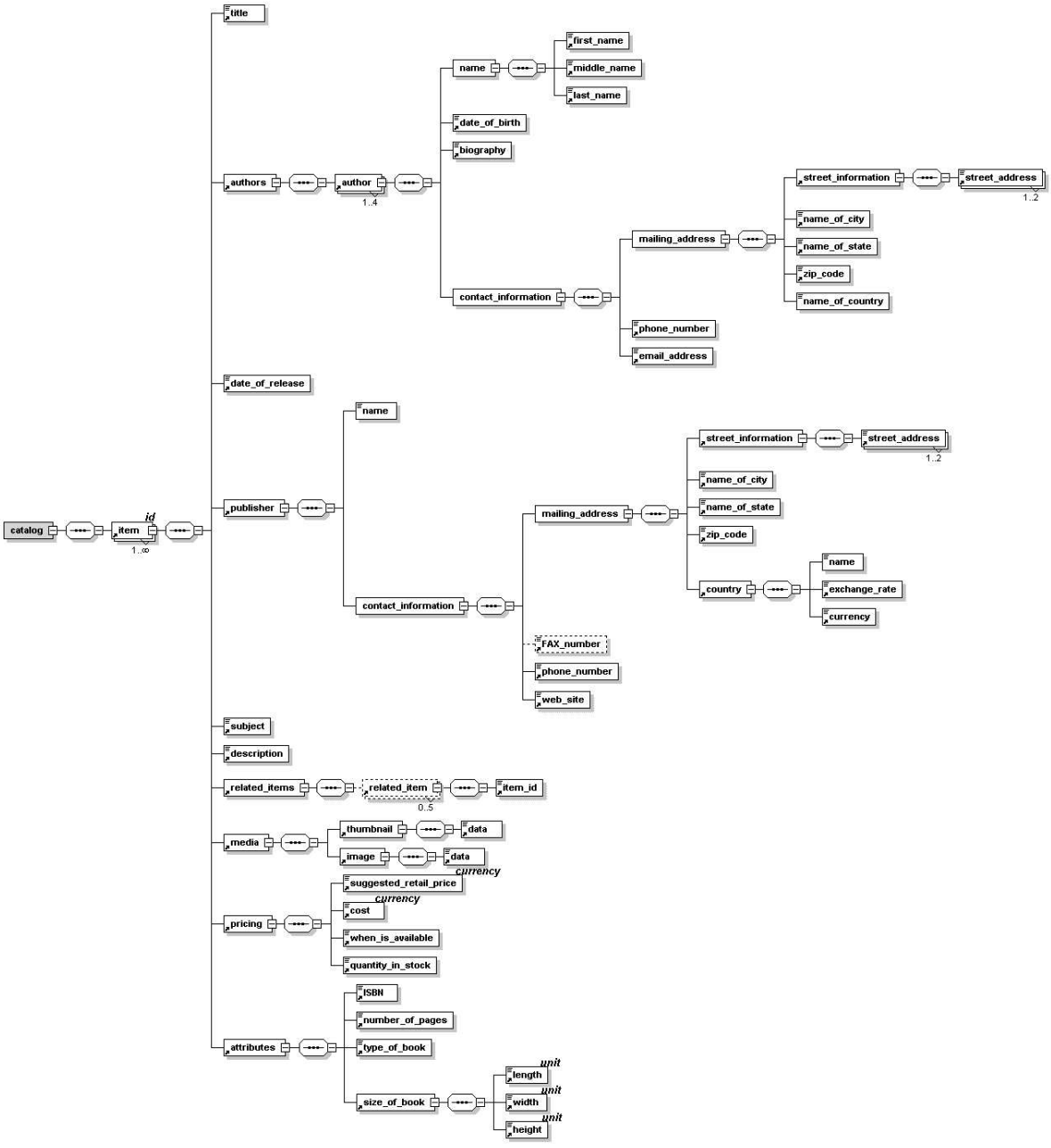

Figure 2.3: Schema Diagram of DC/SD (Catalog)

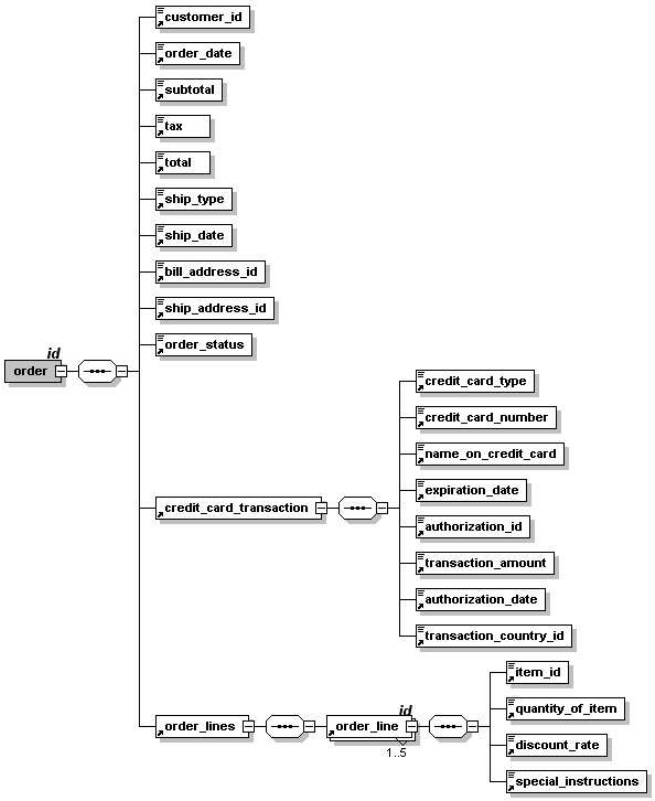

Figure 2.4: Schema Diagram of DC/MD (OrderXXX)

Data-centric multiple documents are transactional and are primarily used for data exchange. Thus, the tags are more descriptive and contain less text content. Usually, the structure is more restricted (in terms of irregularity) and flat (less depth) since most of the data originates in relational databases.

Due to the nature of these documents, we use the *flat translation* (FT) [6, 4] approach that maps a relation into an element type. Each tuple in the relation is mapped into an instance of the element type, and all the columns are mapped into sub-elements (if it is attribute-oriented, all columns are mapped into attributes of the element). The resulting XML documents of this method are very flat. FT approach is used to map five of the TPC-W tables (CUSTOMER, ITEM, AUTHOR, ADDRESS and COUNTRY) to separate XML documents, called *Customer, Item, Author, Address,* and *Country*, respectively. Tables ORDERS, ORDER_LINE and CC_XACTS (tables ORDERS and ORDER_LINE have one-to-many relationship and tables ORDERS and CC_XAVTS have one-to-one relationship), are joined together into one big table and mapped into multiple XML documents called *OrderXXX.xml* (XXX represents the number of a particular document). Each of these documents contains exactly one order information that comes from the three tables.

Figures 2.4-2.9 describe the structures of these XML documents.

2.1.3 Database Generation

For actual data generation, we use ToxGene [1], which is a template-based tool facilitating the generation of synthetic XML documents. A ToXgene template is created for each of the four classes discussed above.

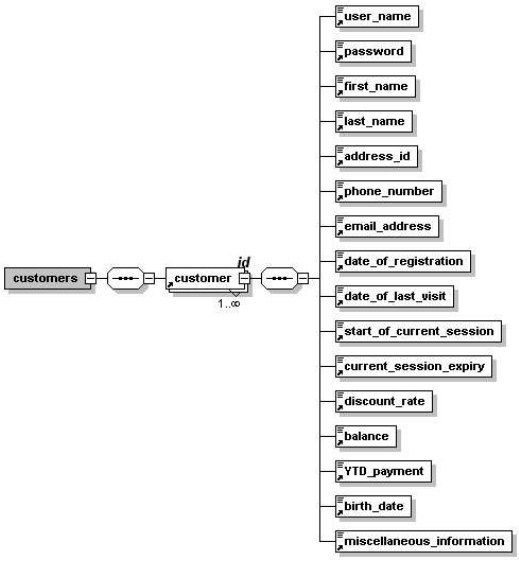

Figure 2.5: Schema Diagram of DC/MD (Customer)

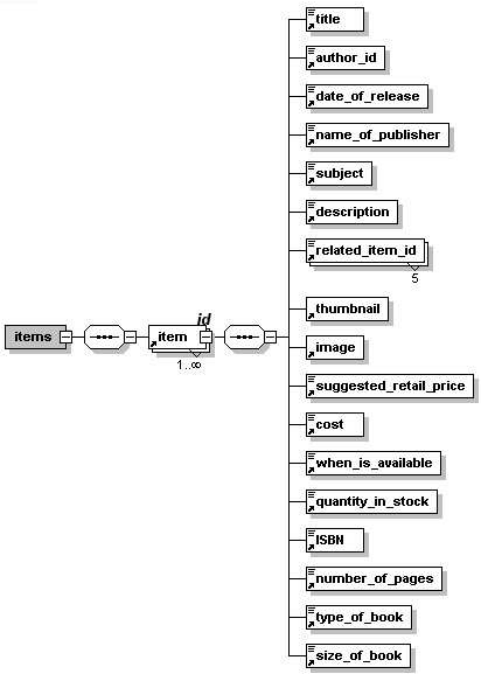

Figure 2.6: Schema Diagram of DC/MD (Item)

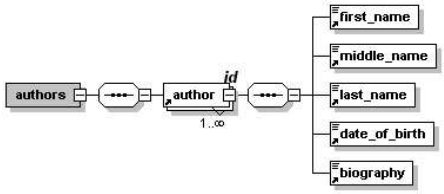

Figure 2.7: Schema Diagram of DC/MD (Author)

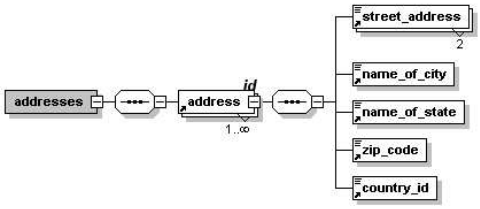

Figure 2.8: Schema Diagram of DC/MD (Address)

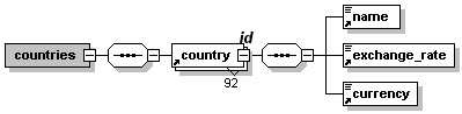

Figure 2.9: Schema Diagram of DC/MD (Country)

2.2 XBench Workload

In this first version of XBench, we only focus on queries and bulk loading; workloads testing update performance will be included in subsequent versions. XBench workload covers the full XQuery functionality as captured in XQuery Use Cases [2]. Queries are classified both along functional categories and in terms of the database type as described in the previous section. In the remainder of this section, a functional overview of the query workload is given where the queries are specified abstractly to demonstrate the functionality that they capture and examples are given from the various document classes[8].

Altogether, there are 20 query types, but each workload class does not contain all of them. XQuery specification of each query is given in [7] and repeated in Appendix A. Readers should consult the database schemas given in the previous section to understand better each of the queries.

- **Exact match.** These queries require string exact match with specified and possibly long path expressions, depending on the levels of predicates being queried in XML documents. Consequently, they can be *shallow* queries, that match only at the top level of XML document trees (example query Q1), or *deep* queries that match the nested structure of XML document tree (query Q2).

  Q1 (DC/SD): *Return the item that has matching item id attribute value X.*

  Q2 (TC/MD): *Find the title of the article authored by Y.*

- **Function application.** These queries challenge the system with aggregate functions such as *count*, *avg*, *max*, *min* and *sum*.

  Q3 (TC/SD): *Group entries by quotation location and calculate the total number of entries in each group.*

- **Ordered access.** These queries test the performance of the system when it preserves the document order during retrieval. This could be *relative order* (Q4) based on the current matching position, or *absolute order* (Q5), which is the order in the document.

  Q4 (TC/MD) *Find the heading of the section following the section entitled "Introduction" in articles written by Y.*

  Q5 (DC/MD) *Return the first order line item of a certain order with id attribute value X.*

- **Quantification.** The two cases are the existentially (Q6) and universally (Q7) quantified queries.

[8] For each example query, its workload class is specified.

Q6 (TC/MD) *Find titles of articles where two keywords "K1" and "K2" are mentioned in the same paragraph.*

Q7 (DC/SD) *Return item information where all its authors are from country Z.*

- **Path expressions.** Two types of queries are defined involving path expressions: (a) queries that contain path expressions where one element name in the path is unknown (Q8), and (b) queries where multiple consecutive element names are unknown (Q9).

  Q8 (TC/SD) *Return quotation text of word "word_1".*

  Q9 (DC/MD) *Return the order status of an order with id attribute value X.*

- **Sorting.** Even though the generic data type of element content in XML documents is string, users may cast the string type to other types. Therefore, these queries test the system in sorting both the string types (Q10) and non-string types (Q11).

  Q10 (DC/MD) *List the orders (order id, order date and ship type), sorted by ship type, ordered within a certain time period.*

  Q11 (TC/SD) *List the quotation authors and quotation dates, sorted by date, for word "word_2".*

- **Document construction.** Structure is important in many XML documents. However, some systems experience difficulties in even preserving the document's original structure. This class of queries test the performance of the system in preserving the structure (Q12) and in transforming the structure (Q13).

  Q12 (DC/SD) *Get the mailing address of the first author of item with id attribute value X.*

  Q13 (TC/MD) *Extract information from the article that has a matching id attribute value Y, including title, the name of the first author, date, and abstract.*

- **Irregular data.** Irregularity of schema is a fact of life in XML databases. These quries test missing elements (Q14) and empty (null) values (Q15).

  Q14 (DC/SD) *Return the names of publishers who publish books in a given time period but do not have a fax number.*

  Q15 (TC/MD) *List author names whose contact elements are empty in articles published within a certain time period.*

- **Retrieval of individual documents.** This query tests an essential function of an XML DBMS to retrieve individual XML documents efficiently while preserving the contents of those documents.

  Q16 (DC/MD) *Retrieve one whole order document with an id attribute value X.*

- **Text search.** These queries test the information retrieval capabilities of systems. Two cases are tested: *uni-gram search* (Q17) where the query contains one particular word, and *bi-gram* and *n-gram search* (Q18) where multiple words are involved.

  Q17 (TC/SD) *Return the headwords of the entries that contain the word "word_x".*

  Q18 (TC/MD) *List the titles and abstracts of articles that contain the phrase "...".*

- **References and joins.** Data-centric documents usually have references to identify the relationship between related data, even among different XML documents. Sometimes users want to combine separate information together using join by values. These queries test this feature.

  Q19 (DC/MD) *For a particular order with id attribute value X, get its customer name and phone, and its order status.*

- **Datatype casting.** The element values in XML documents are String type, but sometimes they need to be cast into other data types.

  Q20 (DC/SD) *Retrieve the item title whose size is larger than a certain number.*

Chapter 3

Mapping of Database

3.1 X-Hive

X-Hive is a Java-based native XML DBMS supporting XQuery. Consequently, there is no need for any database mapping. For bulk loading XML documents, it is possible to use XHAdmin utility. In our experiments, the utility's GUI interface became sluggish after loading the 100M database. We, therefore, used the Java API that X-Hive provides to load the databases.

3.2 DB2

XML Extender is a component included in DB2 to support storage and retrieval of XML documents. There are two options: XML column and XML collection. We use both options for our experiments.

In the case of XML column, an entire XML document is kept as a CLOB in an XML column of a table. A couple of side tables are created for searchable elements/attributes. A column called `dxx_seqno` is also added to each side table in order to keep the ordering information. Data Access Definition (DAD), which is an annotated XML file, is used to define which elements/attributes are searchable. XML column is perfect for application domains where there are multiple small XML documents. It cannot deal with the situation where there is a single large XML document without the help of Text Extender, since the upper limit of a XML CLOB is 2GB. Therefore, in our experiments we use XML column only for DC/MD and TC/MD classes.

XML collection shreds XML documents into one or multiple tables according to DADs explicitly stated by users. The mapping is very similar to the mapping described in Section 2.1.2, except that it is backwards. The DAD file, in this case, is used to describe the mappings from elements/attributes to columns in tables. XML collection shreds a large XML document into multiple tables, so that the system can deal with queries efficiently. We use XML collections for all classes.

The DAD files for all the cases are given in Appendices B.1-B.16.

Consider the XML column DAD for Order (Appendix B.7). All descriptions are inside the element called *xcolumn*. Each XML document is kept intact as CLOB in the table *order_tab*. Other side tables keep data of searchable elements/attributes.

Consider the XML collection DAD for Dictionary (Appendix B.1). All descriptions are inside the element called *xcollection*. The XML document is mapped into 13 tables. The element *condition* is used to describe the (1:n) relationship between those tables. The rest of the mapping file maps individual elements/attributes into columns of each table.

XML Extender can check whether a XML document is well-formed during the loading phase, and does not make use of DTD or XML Schema meta-data to store and index XML documents.

The database designs that are obtained by this mapping for each class are given in Figures 3.1 - 3.7. The detailed DDLs for these relations are given in Appendices C.1 - C.6.

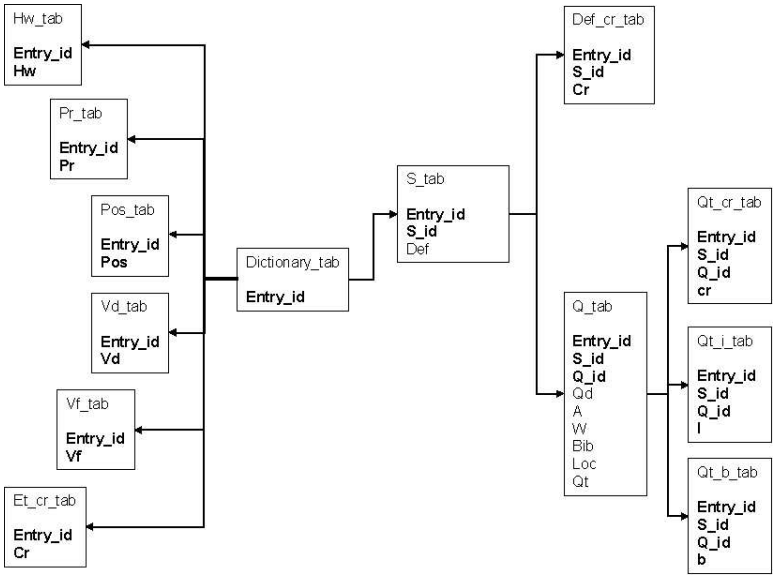

Figure 3.1: Database Schema for TC/SD Class (XML Collection)

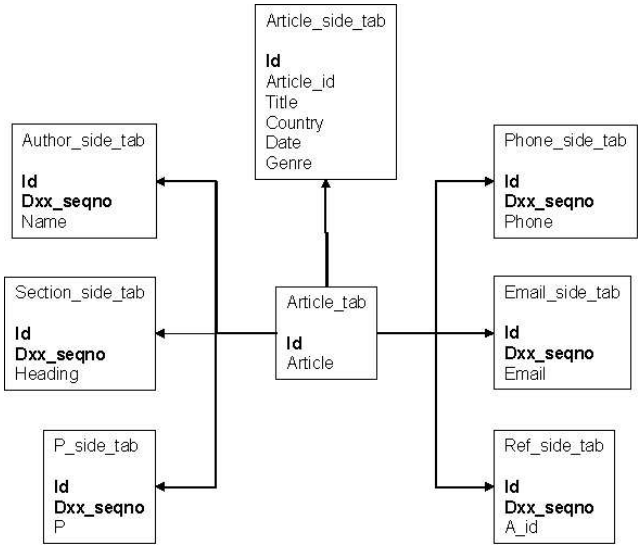

Figure 3.2: Database Schema for TC/MD Class (XML Column)

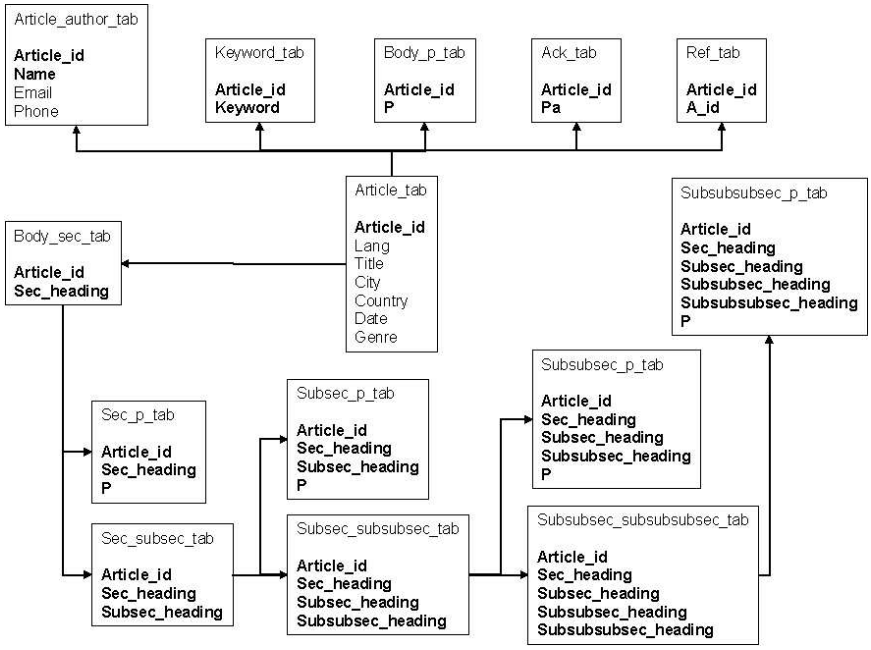

Figure 3.3: Database Schema for TC/MD Class (XML Collection)

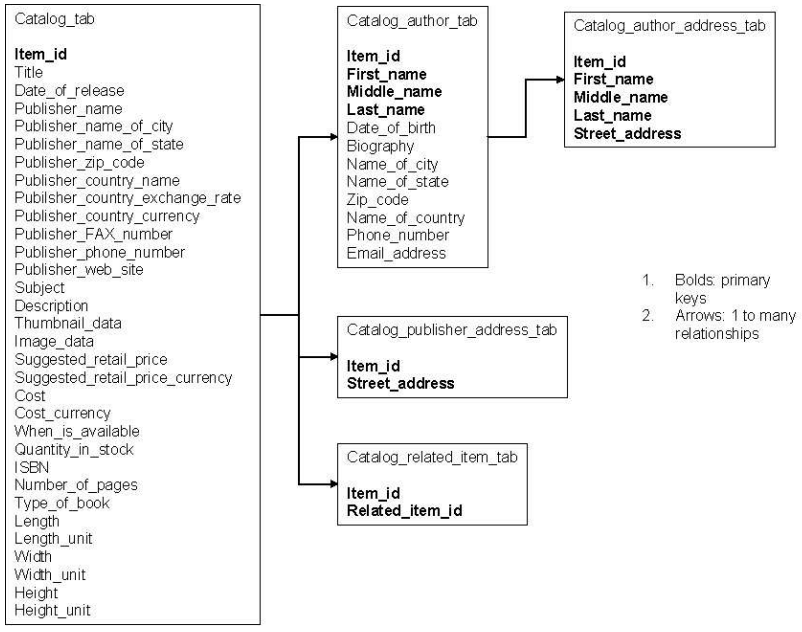

Figure 3.4: Database Schema for DC/SD Class (XML Collection)

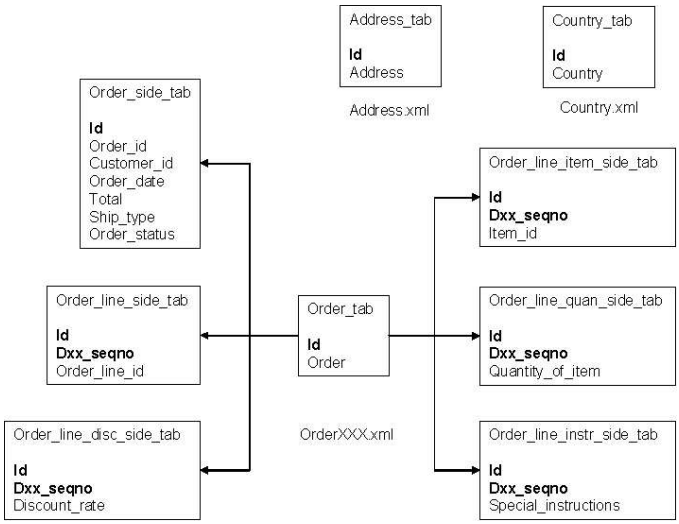

Figure 3.5: Database Schema for DC/MD Class (XML Column)

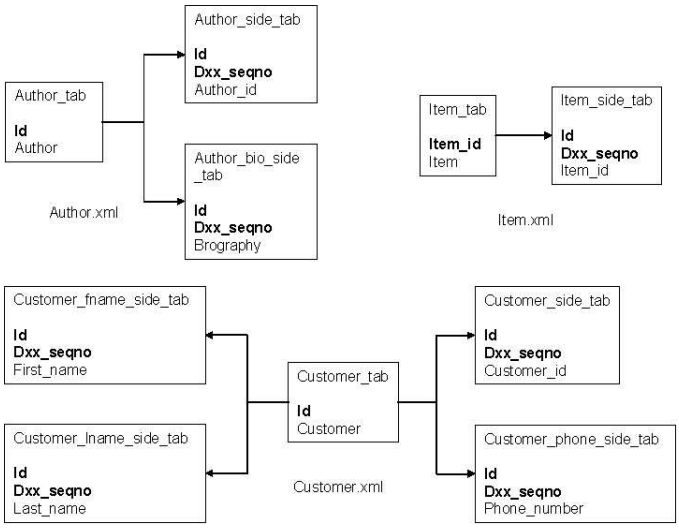

Figure 3.6: Database Schema for DC/MD Class (XML Column) - Continued

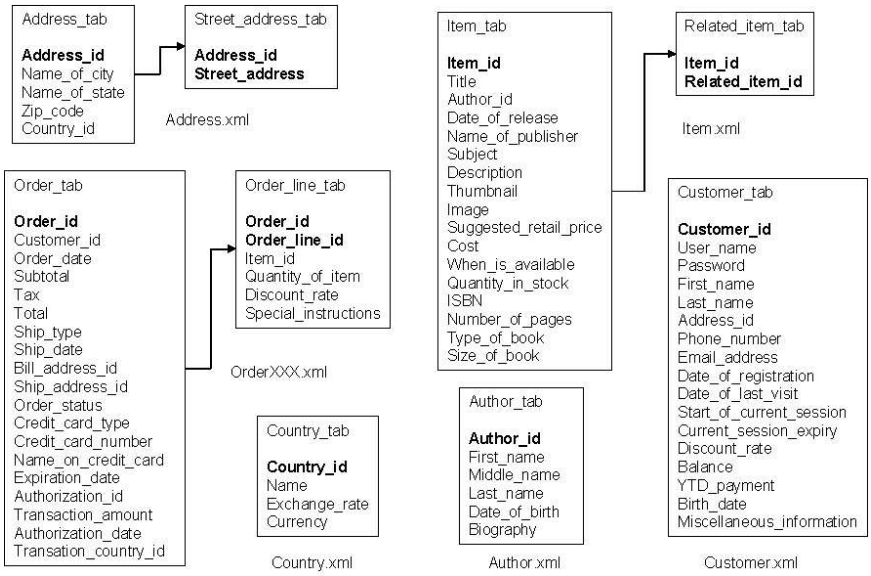

Figure 3.7: Database Schema for DC/MD Class (XML Collection)

3.3 SQL Server

There are two approaches to loading XML data to SQL Server. The first alternative uses stored procedures to load an XML document into main memory and pre-precess it. The document is then shredded and stored in the database. This approach requires the document to be loaded into memory for pre-processing, thus restricting the file size. Many of the documents that are in XBench are too big to be loaded in this manner.

The second approach, which is the one we use in these experiments, is to use the XML bulk loading tool of SQLXML (Version 3.0 SP1) that allows storage of XML documents by shredding them into one or multiple tables. An annotated XML Schema Definition (XSD) can be defined explicitly to describe the mapping, or the mapping can be generated implicitly in the absence of schema information. We use explicit shredding approach in this study. The annotated XSD files for all the cases are given in Appendices D.1-D.9.

Consider the XSD for Order (Appendix D.5). The XML document is mapped into two tables. One table (ORDERS) contains all the basic information about an order and a second table (ORDER_LINES) lists detailed information of order lines. The element `xsd:annotation` in the mapping file basically describes the (1:n) relationship between those two tables. The rest of the mapping file maps individual elements/attributes into columns of each tables. The attribute `sql:relation` declares a table.

A VBScript is used to load XML documents into the SQL Server databases. It is not necessary to create those tables before XML documents are loaded, since they are generated automatically by setting the value of objBL.SchemaGen to true. The VBScript is given below:

```
Dim objBL Set objBL = CreateObject("SQLXMLBulkLoad.SQLXMLBulkload.3.0")

objBL.ConnectionString = "provider=SQLOLEDB.1;datasource=localhost;
database=DCMDS;integrated security=SSPI"

objBL.ErrorLogFile = "c:\error.log" objBL.CheckConstraints=true
objBL.SchemaGen = True objBL.Execute "DCMDOrdShred.xml",
"order1.xml" Set objBL=Nothing
```

SQLXML can check whether a XML document is well-formed during the loading phase, and does not make use of DTD or XML Schema meta-data to store and index XML documents.

The database designs that are obtained by this mapping for each class are given in Figures 3.8 - 3.11[1]. The detailed DDLs for these relations are given in Appendices E.1 - E.4.

[1]Two of these figures, 3.8 and 3.10 are identical to DB2's; we repeat them here for ease of reference.

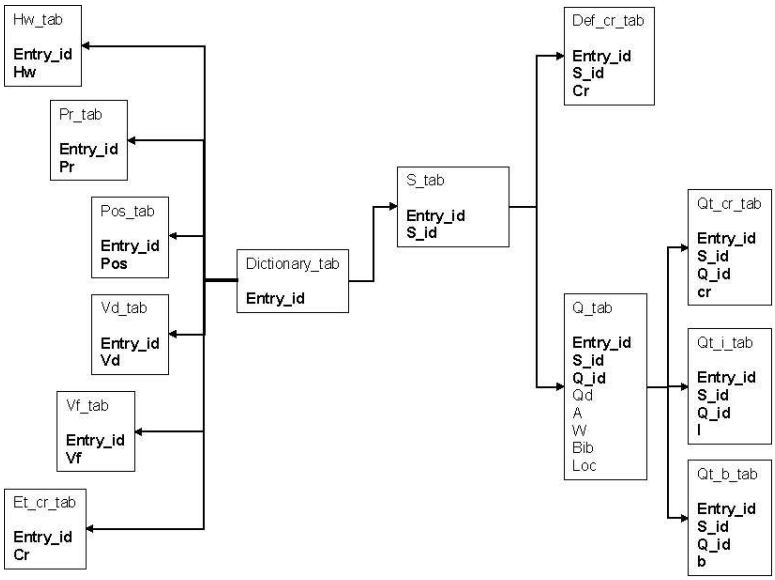

Figure 3.8: Database Schema for TC/SD Class

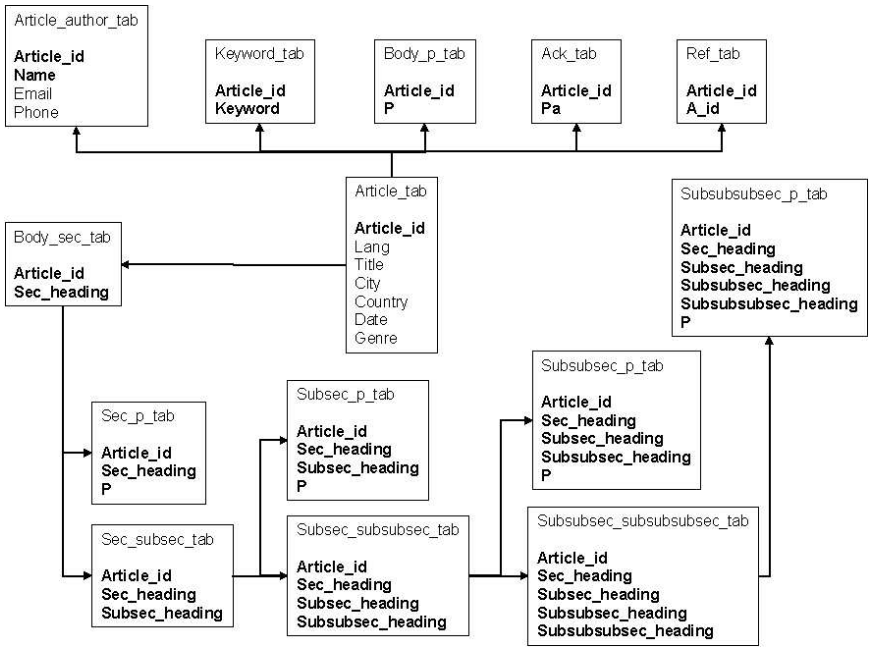

Figure 3.9: Database Schema for TC/MD Class

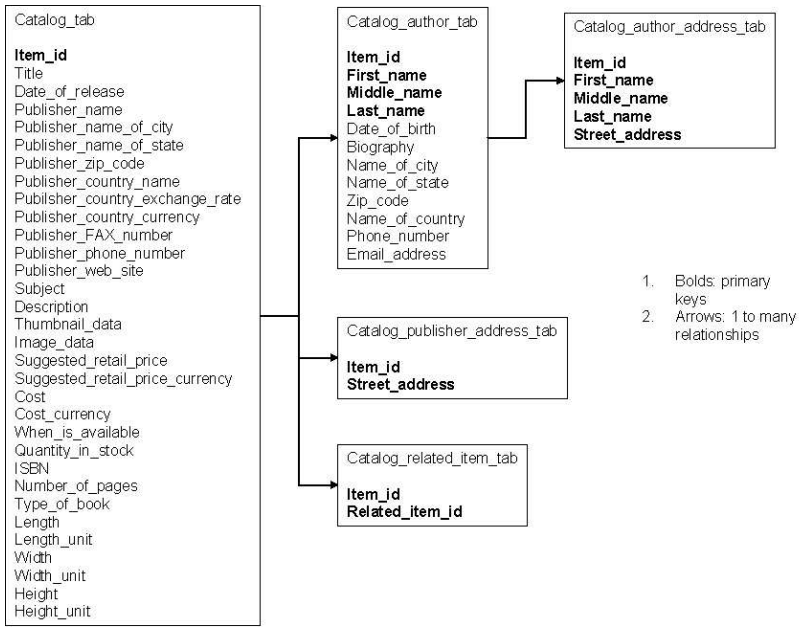

Figure 3.10: Database Schema for DC/SD Class

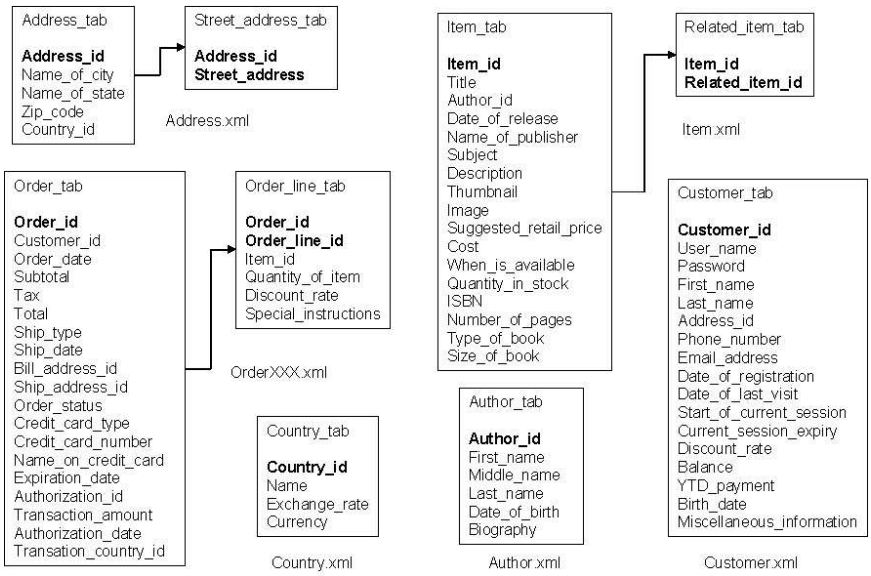

Figure 3.11: Database Schema for DC/MD Class

3.4 Problems with Relational Mappings

There are a number of problems in mapping the XML schemas to relational systems. We discuss these problems below and indicate, in each case, which system has the problem.

1. It is difficult to distinguish between elements and attributes (DB2 XML Collection and SQL Server).
2. It is difficult to deal with and maintain the ordering of child elements (DB2 XML Collection and SQL Server)[2].
3. It is not possible to properly map mixed content elements. We have to ignore these elements with mixed contents, such as the element *qt* in *dictionary.xml* (SQL Server).
4. It is difficult to map chain relationships where the elements on the chain do not have unique values (DB2 XML Collection and SQL Server[3]). For example, consider the following XML document fragment:

```
<root> <article id="1">
   <sec>
     <p>aaaaa</p>
     <p>bbbbbbbb</p>
   </sec>
   <sec>
     <p>cccccccc</p>
     <p>dddddddd</p>
     <p>eeeeeeee</p>
   </sec>
</article> <article id="2">
   <sec>
     <p>ffffff</p>
     <p>ggggggg</p>
   </sec>
   <sec>
     <p>hhhhhh</p>
   </sec>
</article> </root>
```

   Since element `sec` does not have any unique value, it is difficult to keep track of its ordering information, and even worse, it is impossible to tell which `sec` a particular `p` belongs to after the document is shredded. We solve this by adding a unique `id` attribute to those elements that have this problem.

5. Another issue about DB2 XML collection is that the maximum number of rows in a table is restricted to 1024 for a decomposed XML document. This number is fairly small when it comes to big XML documents. Even the document itself is not very big, if there is a leaf element with multiple instances, 1024 will be reached easily. One possible solution is to split the original XML document into smaller documents before loading them into the database. However, during our experiments, we found that a 10 MB XML document almost exceeds this limit. For a 1GB document, we have to split them into approximately 100 files, which is not practical. Therefore, we only have experiments on the databases of size 10 MB for the classes of TC/SD and DC/SD.

[2]It is possible to solve the first two problems by adding extra columns, tables and triggers, but it is not very practical and imposes performance penalties. Therefore, we do not fix these issues in our experiments. Furthermore DB2 XML Column handles order via the `dxx_seqno` column that is added to each side table as discussed in Section 3.2.

[3]In the case of SQL Server the issues related to ordering and chain relationships do not arise if stored procedure approach is used.

6. DB2 XML column solves these issues naturally but suffers from large size XML documents. We only have experiments on the classes of TC/MD and DC/MD.

The end result of these issues is that some of the queries that deal with the issues raised above may not generate correct results, even though we report their performance.

Chapter 4

Performance Results

In these experiments, X-Hive/DB 4.1.1, IBM DB2 UDB 8.1, and Microsoft SQL Server 2000 SP3 are used. All experiments are conducted on a 2Ghz Pentium machine with 1GB main memory and 60GB disk running Windows XP. We report the performance of systems for the small, normal and large database sizes for each database domain (TC/SD, TC/MD, DC/SD, and DC/MD). In our experiments, we create separate database instances for all the scenarios. For example, we create three database instances for TC/SD, called TCSDS (for small), TCSDN (for normal) and TCSDL (for large). The queries are converted to SQL since SQL Server does not support an XQuery interface. The SQL queries, along with the XQuery specifications, are given in Appendix A.

We create indexes on the elements/attributes that are most frequently used by the queries in each document class, and can be implemented in the systems. For example, it is desirable to create indexes on element $< p >$ in TC/MD class. However, neither DB2 nor SQL Server allow the creation of indexes on columns whose lengths are larger than a certain limit (1024 bytes for DB2 and 900 bytes for SQL Server). Consequently, we do not create indexes on element $< p >$ since its length exceeds the limit. In the case of X-Hive, suggestions for indexes were provided by the vendor, along with suggestions for re-writes of some queries. These re-writes are also given in Appendix A.

Table 4.1 shows indexes that are created for each class. Primary/foreign keys of tables are indexed by default if possible, so we do not list them explicitly. In the case of X-Hive, indexes are only created after the documents are loaded into databases. It takes a very long time (around two and half hours) to create full text indexes on text-centric databases with large size (1 GB) and the resulting database files are at least seven times their original sizes. For instance, the sizes of both text-centric database files without indexes are a little bit over 1 GB, but the sizes of the files with indexes are around 9 GB each. Creating full text indexes on data-centric databases are very fast and the sizes of database files with indexes do not increase too much. For example, the size of the large DC/SD database file without indexes is a little bit less than 1 GB and the size with indexes is just 1.1 GB. After consulting with X-Hive, we found out that database files usually grow bigger than what are needed in the end to store indexes and there is a tool called *ootidy* that can clean up database files and reclaim unused disk space. However, the tool itself needs substantial temporary disk space and our experimentation machine does not have enough disk space left to run the tool on large size database files. Therefore, we leave those database files as they are.

As indicated earlier, the results include both those obtained from a database without any indexes and those with indexes. The execution time, in milliseconds, of a query is the cold run time to prevent caching effects.

4.1 Text-Centric Single Document Experiments

The results of this set of experiments for each DBMS are given in Tables 4.2, 4.3 and 4.4. Table 4.5 list the results of all DBMSs with indexes and Table 4.6 list the results of all DBMSs without

Classes	X-Hive		DB2		SQL Server
	Value Index	Full Text	XML Collection	XML Column	
TC/SD	hw, qd, loc, e/@id	e	hw (hw_tab), qd (q_tab), loc (q_tab)	N/A	hw (hw_tab), qd (q_tab), loc (q_tab)
TC/MD	country, date, name, article/@id	p	country (article_tab), date (article_tab), name (article_author_tab)	country (article_side_tab), date (article_side_tab), name (author_side_tab)	country (article_tab), date (article_tab), name (article_author_tab)
DC/SD	date_of_release, first_name, name_of_country, item/@id	description, name	date_of_release (catalog_tab), first_name (catalog_author_tab), name_of_country (catalog_author_tab), description (catalog_tab), publisher_name (catalog_tab)	N/A	date_of_release (catalog_tab), first_name (catalog_author_tab), name_of_country (catalog_author_tab), description (catalog_tab), publisher_name (catalog_tab)
DC/MD	customer_id, discount_rate, total, customer/@id, order/@id	biography	discount_rate (order_line_tab), total (order_tab), biography (author_tab)	discount_rate (order_line_discount_side_tab), total (order_side_tab), biography (author_bio_side_tab)	discount_rate (order_line_tab), total (order_tab), biography (author_tab)

Table 4.1: Indexes for Each Class

indexes[1].

Due to the limits of XML collection (issue 2 in Section 3.4), we can only run experiments on a 10M (small) database. We do not find it feasible to use XML column for this class, so there are no XML column results. Note that Q15 (empty values), Q16 (retrieve individual documents) and Q20 (datatype cast) do not apply for this domain, indicated as N/A in the tables.

[1]In the following tables, we sometimes use these abbreviations in order to save space — S for Small, N for Normal and L for Large.

	TC/SD								
	No Index			No Rewrite but Indexed			Rewrite and Indexed		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	20	1242	13319	10	30	130			
Q2	1231	10555	108566	1763	15402	168222	51	140	6199
Q3	14121	1777756	DNF	511	2814	89599	81	390	25136
Q4	30	2544	DNF	330	1833	7363843	10	10	28260
Q5	20	4547	41359	10	390	7841			
Q6	521	7130	98521	1122	10295	140932	20	80	15773
Q7	30	1382	46447	180	1743	61879			
Q8	250	7101	66876	10	10	85563			
Q9	1002	11827	156986	10	20	20			
Q10	511	7001	101947	50	240	23915			
Q11	20	1242	39848	10	40	120			
Q12	51	1282	48850	10	50	150506			
Q13	20	1452	36753	10	10	158979			
Q14	171	1372	15032	251	2113	73055			
Q15	N/A	N/A	N/A	N/A	N/A	N/A			
Q16	N/A	N/A	N/A	N/A	N/A	N/A			
Q17	711	9023	127974	971	9624	159168	591	2594	87656
Q18	651	8923	139992	982	9784	161733	982	4897	79594
Q19	30	2564	46487	50	1532	77952	40	50	561
Q20	N/A	N/A	N/A	N/A	N/A	N/A			

Table 4.2: X-Hive: Performance of Text-Centric Single Document Workload

	TC/SD (XML Collection)					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	139			80		
Q2	196			70		
Q3	388			100		
Q4	225			90		
Q5	181			85		
Q6	253			150		
Q7	314			150		
Q8	434			70		
Q9	421			70		
Q10	429			55		
Q11	451			60		
Q12	287			85		
Q13	467			75		
Q14	576			55		
Q15	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A
Q17	695			90		
Q18	873			95		
Q19	228			30		
Q20	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.3: DB2: Performance of Text-Centric Single Document Workload

	TC/SD					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	143	1397	9102	90	693	4052
Q2	201	1721	9312	72	294	2195
Q3	401	2395	14092	106	743	4649
Q4	239	1817	9501	93	705	4294
Q5	195	1496	9054	90	594	3754
Q6	261	1447	8345	153	856	6583
Q7	324	1702	10043	154	862	6604
Q8	447	4368	20984	75	436	2537
Q9	438	4295	18465	74	432	2425
Q10	435	3395	15095	57	387	2107
Q11	465	3796	17086	62	401	2532
Q12	295	1687	10034	90	587	3792
Q13	483	2345	12456	85	623	3869
Q14	697	4763	19256	55	353	2256
Q15	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A
Q17	798	9760	47537	95	675	4654
Q18	897	10873	56432	98	689	4713
Q19	243	2304	12496	30	234	1974
Q20	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.4: SQL Server: Performance of Text-Centric Single Document Workload

	TC/SD (Indexed)											
	X-Hive/No Rewrite			X-Hive/Rewrite			XML Collection			SQL Server		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	10	30	130				80			90	693	4052
Q2	1763	15402	168222	51	140	6199	70			72	294	2195
Q3	511	2814	89599	81	390	25136	100			106	743	4649
Q4	330	1833	7363843	10	10	28260	90			93	705	4294
Q5	10	390	7841				85			90	594	3754
Q6	1122	10295	140932	20	80	15773	150			153	856	6583
Q7	180	1743	61879				150			154	862	6604
Q8	10	10	85563				70			75	436	2537
Q9	10	20	20				70			74	432	2425
Q10	50	240	23915				55			57	387	2107
Q11	10	40	120				60			62	401	2532
Q12	10	50	150506				85			90	587	3792
Q13	10	10	158979				75			85	623	3869
Q14	251	2113	73055				55			55	353	2256
Q15	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q17	971	9624	159168	591	2594	87656	90			95	675	4654
Q18	982	9784	161733	982	4897	79594	95			98	689	4713
Q19	50	1532	77952	40	50	561	30			30	234	1974
Q20	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A

Table 4.5: All: Performance of Text-Centric Single Document Workload with Indexes

	TC/SD (Nonindexed)								
	X-Hive			XML Collection			SQL Server		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	20	1242	13319	139			143	1397	9102
Q2	1231	10555	108566	196			201	1721	9312
Q3	14121	1777756	DNF	388			401	2395	14092
Q4	30	2544	DNF	225			239	1817	9501
Q5	20	4547	41359	181			195	1496	9054
Q6	521	7130	98521	253			261	1447	8345
Q7	30	1382	46447	314			324	1702	10043
Q8	250	7101	66876	434			447	4368	20984
Q9	1002	11827	156986	421			438	4295	18465
Q10	511	7001	101947	429			435	3395	15095
Q11	20	1242	39848	451			465	3796	17086
Q12	51	1282	48850	287			295	1687	10034
Q13	20	1452	36753	467			483	2345	12456
Q14	171	1372	15032	576			697	4763	19256
Q15	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q17	711	9023	127974	695			798	9760	47537
Q18	651	8923	139992	873			897	10873	56432
Q19	30	2564	46487	228			243	2304	12496
Q20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.6: All: Performance of Text-Centric Single Document Workload without Indexes

4.2 Text-Centric Multiple Document Experiments

The results of this set of experiments for each DBMS are given in Tables 4.7, 4.8 and 4.9. Table 4.10 list the results of all DBMSs with indexes and Table 4.11 list the results of all DBMSs without indexes.

Q20 (datatype cast) does not apply for this domain.

	TC/MD								
	No Index			No Rewrite but Indexed			Rewrite and Indexed		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	10	10	230	10	10	70			
Q2	140	1312	761	20	40	90			
Q3	51	3014	552835	331	2033	28101			
Q4	10	20	251	10	10	70			
Q5	30	1402	20239	40	40	80			
Q6	270	5858	58895	691	4687	46066	3315	13669	165998
Q7	30	1041	2995	20	240	30504			
Q8	10	20	251	10	10	110			
Q9	30	30	781	50	50	70			
Q10	10	50	1242	40	601	25657			
Q11	20	20	350	10	60	661			
Q12	61	170	400	90	130	180			
Q13	10	20	230	50	70	90			
Q14	20	20	231	50	161	26628			
Q15	60	401	5928	90	561	32808			
Q16	10	10	1552	10	10	110			
Q17	20	120	1532	50	250	23003			
Q18	10	141	2844	30	260	18336			
Q19	21	270	115096	40	50	15091			
Q20	N/A	N/A	N/A	N/A	N/A	N/A			

Table 4.7: X-Hive: Performance of Text-Centric Multiple Document Workload

	TC/MD											
	XML Column						XML Collection					
	Nonindexed			Indexed			Nonindexed			Indexed		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	110	1004	7204	10	10	15	112	1053	7302	10	10	15
Q2	164	1296	7295	15	134	353	167	1302	7312	20	30	45
Q3	386	1897	10483	50	579	5848	390	1921	15401	30	184	3951
Q4	219	1598	7395	40	485	4953	221	1614	7419	35	256	5021
Q5	132	1298	7567	10	10	15	136	1318	7683	20	40	65
Q6	317	1396	7943	174	475	3182	321	1546	9564	295	1395	7843
Q7	391	1574	8032	201	584	4053	401	1768	10094	375	1538	8674
Q8	124	3954	19870	25	187	422	109	3857	17834	10	10	15
Q9	121	3968	19454	25	185	419	120	3927	16975	10	10	15
Q10	395	2845	11466	30	193	2934	396	2871	11592	30	177	3583
Q11	406	3397	12594	35	225	3427	413	3405	12693	25	148	2706
Q12	218	1487	9631	15	20	25	237	1614	9763	70	403	3101
Q13	423	2147	12104	20	51	129	404	2012	11295	30	85	1043
Q14	119	4382	16098	25	477	1950	125	4419	16132	30	165	1685
Q15	211	1798	8497	20	219	982	200	1726	8321	40	228	2492
Q16	197	5935	33967	20	25	30	381	7846	49739	75	497	3387
Q17	115	6649	34287	100	856	7859	130	9654	55854	95	592	4418
Q18	336	11946	62859	296	1860	18404	342	12163	64265	100	619	4578
Q19	196	1846	10687	25	30	35	199	1787	10474	25	35	40
Q20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.8: DB2: Performance of Text-Centric Multiple Document Workload

	TC/MD					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	113	1102	8799	10	10	15
Q2	170	1419	8876	20	35	50
Q3	391	2197	12908	33	193	4003
Q4	226	1809	9432	40	268	5139
Q5	141	1482	9001	20	45	70
Q6	321	1609	10568	304	1412	7923
Q7	401	1794	11056	381	1578	8835
Q8	123	4291	20097	10	10	20
Q9	129	4103	19765	10	10	15
Q10	401	3104	14978	36	191	3654
Q11	429	3594	15012	25	158	2799
Q12	243	1631	9802	80	458	3318
Q13	407	2094	11967	32	95	1102
Q14	125	4696	18989	40	172	1793
Q15	209	1734	8439	45	243	2547
Q16	385	7896	49784	79	514	3496
Q17	125	9729	57295	100	634	4593
Q18	345	12386	65789	104	632	4603
Q19	201	1832	10986	30	40	50
Q20	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.9: SQL Server: Performance of Text-Centric Multiple Document Workload

	TC/MD (Indexed)														
	X-Hive/No Rewrite			X-Hive/Rewrite			XML Column			XML Collection			SQL Server		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	10	10	70				10	10	15	10	10	15	10	10	15
Q2	20	40	90				15	134	353	20	30	45	20	35	50
Q3	331	2033	28101				50	579	5848	30	184	3951	33	193	4003
Q4	10	10	70				40	485	4953	35	256	5021	40	268	5139
Q5	40	40	80				10	10	15	20	40	65	20	45	70
Q6	691	4687	46066	3315	13669	165998	174	475	3182	295	1395	7843	304	1412	7923
Q7	20	240	30504				201	584	4053	375	1538	8674	381	1578	8835
Q8	10	10	110				25	187	422	10	10	15	10	10	20
Q9	50	50	70				25	185	419	10	10	15	10	10	15
Q10	40	601	25657				30	193	2934	30	177	3583	36	191	3654
Q11	10	60	661				35	225	3427	25	148	2706	25	158	2799
Q12	90	130	180				15	20	25	70	403	3101	80	458	3318
Q13	50	70	90				20	51	129	30	85	1043	32	95	1102
Q14	50	161	26628				25	477	1950	30	165	1685	40	172	1793
Q15	90	561	32808				20	219	982	40	228	2492	45	243	2547
Q16	10	10	110				20	25	30	75	497	3387	79	514	3496
Q17	50	250	23003				100	856	7859	95	592	4418	100	634	4593
Q18	30	260	18336				296	1860	18404	100	619	4578	104	632	4603
Q19	40	50	15091				25	30	35	25	35	40	30	40	50
Q20	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.10: ALl: Performance of Text-Centric Multiple Document Workload with Indexes

	TC/MD (Nonindexed)											
	X-Hive			XML Column			XML Collection			SQL Server		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	10	10	230	110	1004	7204	112	1053	7302	113	1102	8799
Q2	140	1312	761	164	1296	7295	167	1302	7312	170	1419	8876
Q3	51	3014	552835	386	1897	10483	390	1921	15401	391	2197	12908
Q4	10	20	251	219	1598	7395	221	1614	7419	226	1809	9432
Q5	30	1402	20239	132	1298	7567	136	1318	7683	141	1482	9001
Q6	270	5858	58895	317	1396	7943	321	1546	9564	321	1609	10568
Q7	30	1041	2995	391	1574	8032	401	1768	10094	401	1794	11056
Q8	10	20	251	124	3954	19870	109	3857	17834	123	4291	20097
Q9	30	30	781	121	3968	19454	120	3927	16975	129	4103	19765
Q10	10	50	1242	395	2845	11466	396	2871	11592	401	3104	14978
Q11	20	20	350	406	3397	12594	413	3405	12693	429	3594	15012
Q12	61	170	400	218	1487	9631	237	1614	9763	243	1631	9802
Q13	10	20	230	423	2147	12104	404	2012	11295	407	2094	11967
Q14	20	20	231	119	4382	16098	125	4419	16132	125	4696	18989
Q15	60	401	5928	211	1798	8497	200	1726	8321	209	1734	8439
Q16	10	10	1552	197	5935	33967	381	7846	49739	385	7896	49784
Q17	20	120	1532	115	6649	34287	130	9654	55854	125	9729	57295
Q18	10	141	2844	336	11946	62859	342	12163	64265	345	12386	65789
Q19	21	270	115096	196	1846	10687	199	1787	10474	201	1832	10986
Q20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.11: All: Performance of Text-Centric Multiple Document Workload without Indexes

4.3 Data-Centric Single Document Experiments

The results of this set of experiments for each DBMS are given in Tables 4.12, 4.13 and 4.14. Table 4.15 lists the results of all DBMSs with indexes and Table 4.16 lists the results of all DBMSs without indexes.

For the same reasons as TC/SD class, we only have results for the small size database using XML collection. Note that Q13 (document structure transforming), Q15 (empty values), Q16 (retrieving individual documents) and Q18 (n-gram search) do not apply for this domain. In the case of X-Hive, we cannot run the original queries Q10, Q11 and Q14 in 1GB DC/SD indexed database. Instead, we get an error message, which is attached in Appendix F, and indicated as ERR in Table 4.12. We were able to run re-written queries Q10, Q11 and Q14 in the same database. Furthermore, X-Hive did not complete execution of some queries within 24 hours and these queries are indicated as DNF in Table 4.12.

	DC/SD								
	No Index			No Rewrite but Indexed			Rewrite and Indexed		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	10	1402	40498	10	20	50			
Q2	161	2604	54518	50	7721	88818	10	10	1402
Q3	2364	698395	DNF	3425	274895	91157698	351	9894	1347527
Q4	3175	5475904	DNF	5538	992978	DNF	90	90	100
Q5	30	2774	35161	10	10	20			
Q6	271	4035	63712	410	4106	60747	40	170	1863
Q7	100	2193	55129	171	2103	26268			
Q8	10	2774	35291	10	20	20			
Q9	20	1542	40949	10	10	51			
Q10	150	3205	58634	530	4296	ERR	210	3004	24685
Q11	90	2103	53477	201	4556	ERR	170	1031	23914
Q12	50	2804	35381	30	50	50			
Q13	N/A	N/A	N/A	N/A	N/A	N/A			
Q14	91	3254	42331	280	22772	ERR	30	180	2674
Q15	N/A	N/A	N/A	N/A	N/A	N/A			
Q16	N/A	N/A	N/A	N/A	N/A	N/A			
Q17	351	4336	49962	311	3284	56301	250	2297	25857
Q18	N/A	N/A	N/A	N/A	N/A	N/A			
Q19	60	1372	85924	90	1602	38355	20	50	420
Q20	160	3194	56471	621	3275	55640			

Table 4.12: X-Hive: Performance of Data-Centric Single Document Workload

	DC/SD (XML Collection)					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	110			50		
Q2	173			10		
Q3	313			60		
Q4	195			40		
Q5	235			10		
Q6	217			60		
Q7	268			65		
Q8	536			15		
Q9	509			10		
Q10	373			30		
Q11	398			30		
Q12	342			20		
Q13	N/A	N/A	N/A	N/A	N/A	N/A
Q14	425			30		
Q15	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A
Q17	496			25		
Q18	N/A	N/A	N/A	N/A	N/A	N/A
Q19	183			20		
Q20	235			230		

Table 4.13: DB2: Performance of Data-Centric Single Document Workload

	DC/SD					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	117	1034	8547	54	79	103
Q2	189	1698	9104	10	10	15
Q3	329	1921	10095	63	495	5284
Q4	208	1706	9239	45	385	4358
Q5	259	1203	7803	15	20	25
Q6	236	1198	6854	64	513	5285
Q7	289	1386	8695	67	567	5826
Q8	560	3905	18093	15	20	25
Q9	528	3760	17095	10	10	10
Q10	386	2969	12956	35	343	2843
Q11	398	2865	14951	34	323	2758
Q12	346	1503	8701	20	25	30
Q13	N/A	N/A	N/A	N/A	N/A	N/A
Q14	456	4523	15386	30	223	2386
Q15	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A
Q17	512	2604	14034	40	304	3194
Q18	N/A	N/A	N/A	N/A	N/A	N/A
Q19	190	1704	9196	20	20	25
Q20	254	1198	7659	250	1198	7659

Table 4.14: SQL Server: Performance of Data-Centric Single Document Workload

	DC/SD (Indexed)											
	X-Hive/No Rewrite			X-Hive/Rewrite			XML Collection			SQL Server		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	10	20	50				50			54	79	103
Q2	50	7721	88818	10	10	1402	10			10	10	15
Q3	3425	274895	91157698	351	9894	1347527	60			63	495	5284
Q4	5538	992978	DNF	90	90	100	40			45	385	4358
Q5	10	10	20				10			15	20	25
Q6	410	4106	60747	40	170	1863	60			64	513	5285
Q7	171	2103	26268				65			67	567	5826
Q8	10	20	20				15			15	20	25
Q9	10	10	51				10			10	10	10
Q10	530	4296	ERR	210	3004	24685	30			35	343	2843
Q11	201	4556	ERR	170	1031	23914	30			34	323	2758
Q12	30	50	50				20			20	25	30
Q13	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q14	280	22772	ERR	30	180	2674	30			30	223	2386
Q15	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q17	311	3284	56301	250	2297	25857	25			40	304	3194
Q18	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A
Q19	90	1602	38355	20	50	420	20			20	20	25
Q20	621	3275	55640				230			250	1198	7659

Table 4.15: All: Performance of Data-Centric Single Document Workload with Indexes

	DC/SD (Nonindexed)								
	X-Hive			XML Collection			SQL Server		
	S	N	L	S	N	L	S	N	L
Q1	10	1402	40498	110			117	1034	8547
Q2	161	2604	54518	173			189	1698	9104
Q3	2364	698395	DNF	313			329	1921	10095
Q4	3175	5475904	DNF	195			208	1706	9239
Q5	30	2774	35161	235			259	1203	7803
Q6	271	4035	63712	217			236	1198	6854
Q7	100	2193	55129	268			289	1386	8695
Q8	10	2774	35291	536			560	3905	18093
Q9	20	1542	40949	509			528	3760	17095
Q10	150	3205	58634	373			386	2969	12956
Q11	90	2103	53477	398			398	2865	14951
Q12	50	2804	35381	342			346	1503	8701
Q13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q14	91	3254	42331	425			456	4523	15386
Q15	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q16	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q17	351	4336	49962	496			512	2604	14034
Q18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q19	60	1372	85924	183			190	1704	9196
Q20	160	3194	56471	235			254	1198	7659

Table 4.16: All: Performance of Data-Centric Single Document Workload without Indexes

4.4 Data-Centric Multiple Document Experiments

The results of this set of experiments for each DBMS are given in Tables 4.17, 4.18 and 4.19. Table 4.20 list the results of all DBMSs with indexes and Table 4.21 list the results of all DBMSs without indexes.

Q2 (deep level exact match), Q13 (document structure transforming), Q15 (empty values), Q18 (n-gram search) and Q20 (data type cast) do not apply for this domain.

	DC/MD								
	No Index			No Rewrite but Indexed			Rewrite and Indexed		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	500	8952	259553	631	23885	4473283			
Q2	N/A	N/A	N/A	N/A	N/A	N/A			
Q3	16513	6094804	DNF	1482	46516	926883	100	19127	215740
Q4	570	54077	DNF	1161	60166	1295839	50	80	941
Q5	340	8692	237531	551	16533	213347			
Q6	371	11076	275253	1162	19587	3329485	491	19157	2956110
Q7	320	10355	295045	1342	21411	862871			
Q8	251	8692	237071	651	16593	957017			
Q9	120	8752	257060	250	8793	221579			
Q10	270	9484	272301	1312	20469	982703	110	6930	176564
Q11	291	9534	275836	1132	18837	857223	80	8062	159339
Q12	110	8432	236871	110	7802	724152			
Q13	N/A	N/A	N/A	N/A	N/A	N/A			
Q14	210	9764	248067	1392	21410	1328701			
Q15	N/A	N/A	N/A	N/A	N/A	N/A			
Q16	210	8743	265151	20	10	310			
Q17	140	8512	249809	901	10165	1494218	50	100	8682
Q18	N/A	N/A	N/A	N/A	N/A	N/A			
Q19	341	18416	538695	60	150	4356			
Q20	N/A	N/A	N/A	N/A	N/A	N/A			

Table 4.17: X-Hive: Performance of Data-Centric Multiple Document Workload

	DC/MD											
	XML Column						XML Collection					
	Nonindexed			Indexed			Nonindexed			Indexed		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	125	1232	8821	10	118	291	129	1235	8832	10	10	15
Q2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q3	379	1835	11943	30	198	1875	383	1845	12003	25	183	2393
Q4	201	1597	8943	10	134	381	214	1745	9287	20	178	2367
Q5	236	1243	7789	90	1598	9567	189	1001	6837	10	10	15
Q6	229	1103	6843	35	128	423	241	1258	7021	164	485	2845
Q7	285	1278	8621	40	154	594	295	1376	8712	184	524	3021
Q8	509	3702	17504	20	454	1870	489	3625	17205	10	10	15
Q9	563	3683	17946	30	1423	7556	512	3598	17345	10	10	15
Q10	376	2795	11965	40	1034	8496	379	2802	11984	176	495	2975
Q11	262	2749	14963	10	345	1296	269	2737	14989	139	428	2485
Q12	342	1497	8813	30	1487	7631	324	1326	8709	10	10	15
Q13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q14	457	3198	15096	10	143	398	487	3386	16242	50	1343	12432
Q15	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q16	369	1357	9932	30	164	748	359	1280	9920	30	219	2596
Q17	486	35870	148450	10	8649	54287	457	2476	13045	20	187	1754
Q18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q19	216	1892	11048	30	1586	8739	209	1707	10296	30	230	2645
Q20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.18: DB2: Performance of Data-Centric Multiple Document Workload

	DC/MD					
	Nonindexed			Indexed		
	Small	Normal	Large	Small	Normal	Large
Q1	139	1356	8976	10	10	15
Q2	N/A	N/A	N/A	N/A	N/A	N/A
Q3	389	2069	12045	25	195	2501
Q4	214	1795	9369	20	189	2473
Q5	287	1305	8706	10	10	20
Q6	247	1297	7139	178	506	2986
Q7	301	1406	8875	198	543	3172
Q8	598	4059	19483	10	10	20
Q9	578	3905	18697	10	10	15
Q10	398	3002	13985	180	513	3099
Q11	387	2987	15194	142	444	2595
Q12	367	1605	8905	10	10	20
Q13	N/A	N/A	N/A	N/A	N/A	N/A
Q14	495	4693	16875	193	1520	14318
Q15	N/A	N/A	N/A	N/A	N/A	N/A
Q16	384	1607	10021	35	241	2729
Q17	597	6705	15879	55	216	1918
Q18	N/A	N/A	N/A	N/A	N/A	N/A
Q19	229	1907	11296	35	246	2718
Q20	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.19: SQL Server: Performance of Data-Centric Multiple Document Workload

	DC/MD (Indexed)														
	X-Hive/No Rewrite			X-Hive/Rewrite			XML Column			XML Collection			SQL Server		
	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large	Small	Normal	Large
Q1	631	23885	4473283				10	118	291	10	10	15	10	10	15
Q2	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q3	1482	46516	926883	100	19127	215740	30	198	1875	25	183	2393	25	195	2501
Q4	1161	60166	1295839	50	80	941	10	134	381	20	178	2367	20	189	2473
Q5	551	16533	213347				90	1598	9567	10	10	15	10	10	20
Q6	1162	19587	3329485	491	19157	2956110	35	128	423	164	485	2845	178	506	2986
Q7	1342	21411	862871				40	154	594	184	524	3021	198	543	3172
Q8	651	16593	957017				20	454	1870	10	10	15	10	10	20
Q9	250	8793	221579				30	1423	7556	10	10	15	10	10	15
Q10	1312	20469	982703	110	6930	176564	40	1034	8496	176	495	2975	180	513	3099
Q11	1132	18837	857223	80	8062	159339	10	345	1296	139	428	2485	142	444	2595
Q12	110	7802	724152				30	1487	7631	10	10	15	10	10	20
Q13	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q14	1392	21410	1328701				10	143	398	50	1343	12432	193	1520	14318
Q15	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q16	20	10	310				30	164	748	30	219	2596	35	241	2729
Q17	901	10165	1494218	50	100	8682	10	8649	54287	20	187	1754	55	216	1918
Q18	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q19	60	150	4356				30	1586	8739	30	230	2645	35	246	2718
Q20	N/A	N/A	N/A				N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.20: All: Performance of Data-Centric Multiple Document Workload with Indexes

	DC/MD (Nonindexed)											
	X-Hive			XML Column			XML Collection			SQL Server		
	S	N	L	S	N	L	S	N	L	S	N	L
Q1	500	8952	259553	125	1232	8821	129	1235	8832	139	1356	8976
Q2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q3	16513	6094804	DNF	379	1835	11943	383	1845	12003	389	2069	12045
Q4	570	54077	DNF	201	1597	8943	214	1745	9287	214	1795	9369
Q5	340	8692	237531	236	1243	7789	189	1001	6837	287	1305	8706
Q6	371	11076	275253	229	1103	6843	241	1258	7021	247	1297	7139
Q7	320	10355	295045	285	1278	8621	295	1376	8712	301	1406	8875
Q8	251	8692	237071	509	3702	17504	489	3625	17205	598	4059	19483
Q9	120	8752	257060	563	3683	17946	512	3598	17345	578	3905	18697
Q10	270	9484	272301	376	2795	11965	379	2802	11984	398	3002	13985
Q11	291	9534	275836	262	2749	14963	269	2737	14989	387	2987	15194
Q12	110	8432	236871	342	1497	8813	324	1326	8709	367	1605	8905
Q13	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q14	210	9764	248067	457	3198	15096	487	3386	16242	495	4693	16875
Q15	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q16	210	8743	265151	369	1357	9932	359	1280	9920	384	1607	10021
Q17	140	8512	249809	486	35870	148450	457	2476	13045	597	6705	15879
Q18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q19	341	18416	538695	216	1892	11048	209	1707	10296	229	1907	11296
Q20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 4.21: All: Performance of Data-Centric Multiple Document Workload without Indexes

4.5 Discussion of Experimental Results

4.5.1 About X-Hive

X-Hive does not perform well when the database size is large (1GB). Without indexes, in most classes, Q3 (function application, *group by*) and Q4 (related ordered access) cannot even finish within 24 hours. With index but no re-write of queries, the situation is somewhat better but there is no substantial change. With index and re-written queries, the performance is much better.

The following are the major observations:

- The positive effect of indexes do not reveal themselves until the database sizes get large. Sometimes, the performance of indexed databases are even worse than non-indexed databases for small database sizes.
- In most cases, if we do not re-write queries (specific to X-Hive) that fully take advantage of indexes, even though indexes are there, the results may be even worse than the cases without indexes[2]. The use of *unordered* is a good example – consider the results of query Q2 in TC/SD class. The performance of the original query with indexes is worse than the query without indexes. However, the results of the re-write query are much better. The explanation provided by the vendor is that X-Hive does not return index lookup results in document order. Hence, if a large result set from an index lookup needs to be sorted afterwards, this will sometimes completely cancel the benefits of using the index. In such cases the "unordered" keyword will do wonders. However, query semantics may be slightly changed, as the order of the final result set may be different from the result set of the original query.
- Full text indexing is not as efficient as value indexing. The performance improvements of queries with full text indexes are not as obvious as those of queries with value indexes. We are told by the vendor that this is because, in the case of value indexing, the whole value is regarded as the key, so there are usually fewer keys and each item can only be found under one key.
- For the DC/MD class, creating indexes does not help improve performance of most queries. This is likely due to the large number of files that are created.

4.5.2 Document Structure and Its Physical Storage

XML enabled DBMSs store XML documents using one of the following three approaches: storing an XML document as a whole, shredding an XML document into fragments, and combining the above two approaches. The first approach works efficiently on ordered access and retrieving big fragments or whole documents, but performs poorly on text search. The second approach works well on text search but is problematic on ordered access and document reconstruction. The last approach suffers high space cost and redundancy. Queries such as Q5, Q12 and Q17 are designed to investigate this issue.

The ordering of elements is important in XML documents. Although XML documents are stored in relational DBMSs, the orders should still be preserved. Query Q5 is designed to return data based on its order in a document.

One of the big problems experienced by relational DBMSs in storing XML documents is their poor performance on document reconstruction. Depending on the approach used in storing XML documents in DBMSs, both relational DBMSs cannot even preserve the document's original structure in some cases. However, structure is very important to XML documents, especially to text documents. Query Q12 retrieves fragments of original documents with original structures.

Text search plays a very important part in XML document systems. The integration of information retrieval (IR) technologies with database querying is an emerging area of study. Regardless

[2]There is one exception. The performance of the rewrite query Q6 for TC/MD class is worse than that of the original query Q6.

of the storage mechanism that a DBMS uses, the system should perform well on uni-gram, bi-gram and n-gram search. Query Q17 returns XML documents that contain a particular word.

Two relational systems take advantage of the indexes on primary/foreign keys to speed up execution of query Q5. DB2/XML Column can keep track of ordering information by using `dxx_seqno`. DB2/XML Collection and SQL Server happen to return correct results for this particular query but they do not guarantee correctness since they do not maintain document order nor do they store any ordering information.

X-Hive performs very well in XML document construction (Q12); it produces the correct result in a short time. Other systems' responses are relatively slow. Since XML documents are shredded and stored in tables, numerous joins are required in order to re-construct the document. Even worse, the structure of the resulting document fragment is not necessarily the same as the original. DB2/XML Column is an exception because it stores XML documents intact.

For Q5 and Q12, creating necessary indexes on some commonly used elements speeds up the queries, specially for X-Hive, since two relational DBMSs automatically create some indexes on primary/foreign keys when bulk loading, which may (most likely) cover those elements. The indexing does not make a big difference for small databases, but start to take positive effects when the databases get larger.

None of the systems does well on Q17 that involves text search. X-Hive supports the creation of full text indexes on particular elements. We do not use it in our experiments because we cannot build similar full text indexes for the relational systems. Use of full-text index will certainly improve its performance. For relational DBMSs, when elements are shredded into many tables, creating full-text indexes on these elements means creating indexes on all columns in the tables. Furthermore, DB2 and SQL Server have limits for indexing on the size of columns and most likely those columns are too big to create indexes on. IBM has a different product, called Text Extender, which is targeted to this issue, but this is not used in our experiments.

Overall, in this experiment, DB2/XML Collection and SQL Server perform better than X-Hive in data-centric and text-centric/single document domains, especially in large database sizes (beyond 100MB). X-Hive and DB2/XML Column do better in text-centric/multiple document domain. DB2/XML Collection does slight better than SQL server.

Based on the results, as expected, relational DBMSs have a more optimized storage mechanism, particularly for large tables. Since we map multiple *orderXXX.xml* documents into two tables (*order_tab* and *order_line_tab*) in DC/MD case, relational systems can take advantage of the efficient storage mechanism. On the other hand, X-Hive suffers from accessing huge amounts of XML documents in DC/MD case. For TC/MD domain, there is a limited number of small documents, and that is where X-Hive shows its advantage.

X-Hive also outperforms the other two systems when it comes to reconstruction of document fragments in text-centric domains if original XML documents are not very big (less than 1GB). It should be noted that since the mapping does not maintain the document order or the mixed content elements, the results returned by those two relational DBMSs for Q5 and Q12 are not necessarily accurate.

4.5.3 Path Expression and Loose Schema

Queries such as Q8 and Q14 focus on measuring the performance of accessing XML documents using path expressions and dealing with the irregularity of XML documents.

Even though some relational DBMSs do not directly support path expressions, they should be able to process them after the expressions are translated into SQL [3] or system specific languages. It is common that one or more elements in a path expression are unknown, and query Q8 tests this case.

Sometimes the elements in a path expression are not fully known or elements in an XML document are missing. This is due to the fact that XML documents do not have as strict schemas as their relational counterparts. XML schemas are more flexible and may have a number of irregularities such as missing elements and empty values. It is a challenge for relational DBMSs to store such

loose schema data. Smart DBMSs should be able to exploit XML schemas associated with XML documents in dealing with irregularity. Query Q14 covers missing elements in XML documents case.

The results show that, in most cases, relational DBMSs do well on Q8. Recall that in those cases, XQuery are mapped into SQL, and hence, no real path expressions are actually involved in executing the query. The results of these two queries are very similar. X-Hive, on that hand, needs to deal with path expressions and its execution times are close to those of the relational systems.

Both relational DBMSs suffer from table scanning for Q14, due to the fact that we do not purposely create any index on that particular missing element, while X-Hive does pretty well in some cases even though there is no index.

X-Hive outperforms DB2 and SQL Server in text-centric categories but performs badly on large size databases of data-centric domains which are the strong points of DB2 and SQL Server.

4.6 Complexity of Mapping and Bulk Loading Time

The efficiency of loading XML documents into each DBMS also draws our attention. For relational DBMSs, there is the additional issue of mapping XML into relational tables and the effect that this would have on the bulk loading time. We turn off the option of conforming/validating XML documents (if a DBMS provides such functionality) during bulk loading to reduce time costs. Our results that are grouped by database type and then by database size are reported in Table 4.22.

	DC/SD			DC/MD			TC/SD			TC/MD		
	S	N	L	S	N	L	S	N	L	S	N	L
X-Hive	9	59	517	25	304	8568	12	72	647	7	57	512
XML Column	-	-	-	30	417	11532	-	-	-	12	85	662
XML Collection	34	-	-	87	1126	31860	46	-	-	40	124	762
SQL Server	43	120	770	119	1438	39496	55	153	960	52	148	894

Table 4.22: Bulk Loading Time in Seconds (S:Small, N:Normal, L:Large)

As expected, the bulk loading times of two relational DBMSs are much longer than that of X-Hive because they need extra time to shred documents and create indexes for primary/foreign keys. Data-centric documents generally are loaded faster than text-centric documents due to the relative simplicity of data-centric document structures. For a given database size, the bigger the individual files, the slower is the loading time. However, the number of documents becomes very critical once the number becomes very high. All DBMSs load data much slower in DC/MD than others for the same data size, due to the fact that DC/MD has many more XML documents than the others, requiring extra I/O operations. When the database size increases 10 times, the loading time does not necessarily increase linearly except for DC/MD, in which case, the number of files (e.g., over 200k for 1GB database size) dominates the loading time.

Bibliography

[1] D. Barbosa, A. Mendelzon, J. Keenleyside, and K. Lyons. ToXgene: A Template-Based Data Generator for XML. In *Proc. 4th Int. Workshop on the Web and Databases (WebDB)*, pages 49–54, June 2002.

[2] D. Chamberlin, P. Fankhauser, M. Marchiori, and J. Robie. XML Query Use Cases. http://www.w3.org/TR/xmlquery-use-cases.

[3] D. DeHaan, D. Toman, M. Consens, and M. T. Özsu. A Comprehensive XQuery to SQL Translation Using Dynamic Interval eEncoding. In *ACM SIGMOD Int. Conf. on Management of Data*, page 623-634, June 2003.

[4] D. Lee, M. Mani, F. Chiu, and W.W. Chu. Nesting-Based Relational-to-XML Schema Translation. In *Proc. 4th Int. Workshop on the Web and Databases (WebDB)*, pages 61–66, May 2001.

[5] D. Lee, M. Mani, F. Chiu, and W.W. Chu. Effective Schema Conversions between XML and Relational Models. In *Proc. European Conf. on Artificial Intelligence (ECAI), Knowledge Transformation Workshop*, July 2002.

[6] V. Turau. Making Legacy Data Accessible for XML Applications. http://www.informatik.fh-wiesbaden.de/∼turau/veroeff.html, 1999.

[7] B. B. Yao, M. T. Özsu, and J. Keenleyside. XBench - A Family of Benchmarks for XML DBMSs. Technical Report CS-TR-2002-39, University of Waterloo, School of Computer Science, Waterloo, Canada, December 2002. Available from http://db.uwaterloo.ca/∼ddbms/projects/xbench.

Appendix A

XBench Workload Queries and their SQL Equivalents

This appendix lists the full set of queries, in XQuery specification and in their SQL equivalents for each DBMS. Since X-Hive that we use for experiments does not support some features of the latest XQuery specification, we will give equivalent queries in the version that X-Hive support as well, if they are different from original queries. Most of queries are the same except for Q10 and Q11, where the current XQuery specification supports *order by* but X-Hive uses *sort by* instead. Some queries for X-Hive are also rewritten for indexed databases in order to improves the performance. The queries are categorized with respect to the workload classes as identified by the database type.

A.1 Text-Centric Single Document

The following queries are based on an implicit (unnamed) input data set, which is "dictionary.xml".

- **TC/SD_Q1** *Return the entry that has matching headword ("the").*

  XQuery:

```
for $ent in input()/dictionary/e
where $ent/hwg/hw="the"
return
    $ent
```

  XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
```

```
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'the'
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'the'
for xml auto, elements
```

- **TC/SD_Q2** *Find the headword of the entry which has matching quotation year (1900).*

  XQuery:

  ```
  for $ent in input()/dictionary/e
  where $ent/ss/s/qp/q/qd="1900"
  return
      $ent/hwg/hw
  ```

  X-Hive/Rewrite:

  ```
  unordered for $ent in input()/dictionary/e
  where $ent/ss/s/qp/q/qd="1900"
  return
  $ent/hwg/hw
  ```

XML Collection:

```
select hw from hw_tab,
    s_tab,
    q_tab
where hw_tab.entry_id = s_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    q_tab.qd = '1900'
```

SQL Server:

```
select hw
from hw_tab,
    s_tab,
    q_tab
where hw_tab.entry_id = s_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    q_tab.qd = '1900'
for XML auto
```

- **TC/SD_Q3** *Group entries by quotation location in a certain quotation year (1900) and calculate the total number entries in each group.*

XQuery:

```
for $a in distinct-values
    (input()/dictionary/e/ss/s/qp/q[qd="1900"]/loc)
let $b := input()/dictionary/e/ss/s/qp/q[loc=$a]
return
    <Output>
        <Location>{$a/text()}</Location>
        <NumberOfEntries>{count($b)}</NumberOfEntries>
    </Output>
```

X-Hive/Rewrite:

```
unordered for $a in distinct-values
(input()/dictionary/e/ss/s/qp/q[qd="1900"]/loc)
let $b := unordered input()/dictionary/e/ss/s/qp/q[loc=$a]
return
<Output>
  <Location>
    {$a/text()}
  </Location>
  <NumberOfEntries>
    {count($b)}
  </NumberOfEntries>
</Output>
```

XML Collection:

```
select loc,
    count(distinct(entry_id)) number
from q_tab
where qd = '1900'
group by loc
```

SQL Server:

```
select *
from
(select loc,
    count(distinct(entry_id)) number
from q_tab
where qd = '1900'
group by loc) temp
for xml auto
```

- **TC/SD_Q4** *List the headword of the previous entry of a matching headword ("you").*

  XQuery:

```
let $ent := input()/dictionary/e[hwg/hw="you"]
for $prevEnt in input()/dictionary/e[hwg/hw << $ent]
    [position() = last()]
return
    <Output>
        <CurrentEntry>{$ent/hwg/hw/text()}</CurrentEntry>
        <PreviousEntry>{$prevEnt/hwg/hw/text()}</PreviousEntry>
    </Output>
```

  X-Hive/Rewrite:

```
let $ent := input()/dictionary/e[hwg/hw="you"]
for $prevEnt in $ent/preceding::e[hwg/hw][1]
return
<Output>
  <CurrentEntry>
    {$ent/hwg/hw/text()}
  </CurrentEntry>
  <PreviousEntry>
    {$prevEnt/hwg/hw/text()}
  </PreviousEntry>
</Output>
```

  XML Collection:

```
select curr, hw_tab.hw from (select t1.hw curr,
    max(t2.entry_id) prev
from hw_tab t1,
    hw_tab t2
where t1.hw = 'you' and
    t1.entry_id > t2.entry_id
group by t1.hw) temp,
    hw_tab
where hw_tab.entry_id = temp.prev
```

  SQL Server:

```
select curr, hw_tab.hw
from
(select t1.hw curr,
    max(t2.entry_id) prev
from hw_tab t1,
    hw_tab t2
where t1.hw = 'you' and
    t1.entry_id > t2.entry_id
```

```
group by t1.hw) temp,
    hw_tab
where hw_tab.entry_id = temp.prev
```

- **TC/SD_Q5** *Return the first sense of a matching headword ("that").*

  XQuery:

```
for $a in input()/dictionary/e
where $a/hwg/hw="that"
return
    $a/ss/s[1]
```

  XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'that'
```

  SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'that'
for xml auto, elements
```

- **TC/SD_Q6** *Return the words where some quotations were quoted in a certain year (1900).*

  XQuery:

```
for $word in input()/dictionary/e
where some $item in $word/ss/s/qp/q
    satisfies $item/qd eq "1900"
return
    $word
```

X-Hive/Rewrite:

```
unordered for $word in input()/dictionary/e
where $word/ss/s/qp/q/qd[. = "1900"]
return
$word
```

XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    exists
    (select t1.entry_id
    from q_tab t1
    where t1.entry_id = dictionary_tab.entry_id and
    qd = '1900')
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
```

```
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    exists
    (select t1.entry_id
    from q_tab t1
    where t1.entry_id = dictionary_tab.entry_id and
    qd = '1900')
for xml auto, elements
```

- **TC/SD_Q7** *Return the words where all quotations were quoted in a certain year (1900).*

  XQuery:

```
for $word in input()/dictionary/e
where every $item in $word/ss/s/qp/q
    satisfies $item/qd eq "1900"
return
    $word
```

  XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
```

```
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    not exists
    (select t1.entry_id
    from q_tab t1
    where t1.entry_id = dictionary_tab.entry_id and
    qd <> '1900')
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    vd_tab,
    vf_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = vd_tab.entry_id and
    dictionary_tab.entry_id = vf_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    not exists
    (select t1.entry_id
    from q_tab t1
    where t1.entry_id = dictionary_tab.entry_id and
    qd <> '1900')
for xml auto, elements
```

- **TC/SD_Q8** *Return Quotation Text (one element name unknown) of a word ("and").*

  XQuery:

```
for $ent in input()/dictionary/e
where $ent/*/hw = "and"
return
    $ent/ss/s/qp/*/qt
```

  XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where   dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'and'
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where   dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'and'
for xml auto, elements
```

- **TC/SD_Q9** *Return Quotation Text (several consecutive element names unknown) of a word ("and").*

XQuery:

```
for $ent in input()/dictionary/e
where $ent//hw = "or"
return
    $ent//qt
```

XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where   dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'or'
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where   dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'or'
for xml auto, elements
```

- **TC/SD_Q10** *List the words and their pronunciation, alphabetically, quoted in a certain year (1900).*

  XQuery:

```
for $a in input()/dictionary/e
where $a/ss/s/qp/q/qd = "1900"
order by $a/hwg/hw
return
    <Output>
        {$a/hwg/hw}
        {$a/hwg/pr}
    </Output>
```

  X-Hive:

```
for $a in input()/dictionary/e
where $a/ss/s/qp/q/qd = "1900"
return
    <Output>
        {$a/hwg/hw}
        {$a/hwg/pr}
    </Output>
sort by (hw)
```

  XML Collection:

```
select hw,pr from hw_tab,
    pr_tab,
    s_tab,
    q_tab
where hw_tab.entry_id = s_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = pr_tab.entry_id and
    q_tab.qd = '1900'
order by hw
```

  SQL Server:

```
select hw,pr
from hw_tab,
    pr_tab,
    s_tab,
```

```
    q_tab
where hw_tab.entry_id = s_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = pr_tab.entry_id and
    q_tab.qd = '1900'
order by hw
for XML auto, elements
```

- **TC/SD_Q11** *List the quotation locations and quotation dates, sorted by date, for a word ("word").*

  XQuery:

```
for $a in input()/dictionary/e
    [hwg/hw="the"]/ss/s/qp/q
order by $a/qd
return
    <Output>
        {$a/a}
        {$a/qd}
    </Output>

for $a in input()/dictionary/e
    [hwg/hw="the"]/ss/s/qp/q
return
    <Output>
        {$a/a}
        {$a/qd}
    </Output>
sort by (qd)
```

  XML Collection:

```
select a,
    qd
from hw_tab,
    q_tab
where hw_tab.entry_id = q_tab.entry_id and
    hw_tab.hw = 'word'
order by qd
```

  SQL Server:

```
select a,
    qd
from hw_tab,
    q_tab
where hw_tab.entry_id = q_tab.entry_id and
    hw_tab.hw = 'word'
order by qd
for xml auto, elements
```

- **TC/SD_Q12** *Retrieve the senses of a word ("his").*

  XQuery:

```
for $a in input()/dictionary/e
where $a/hwg/hw="his"
```

```
return
    <Entry>
        {$a/ss}
    </Entry>
```

XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'his'
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'his'
for xml auto, elements
```

- **TC/SD_Q13** *Construct a brief information on a word ("his"), including: headword, pronunciation, part_of_speech, first etymology and first sense definition.*

XQuery:

```
for $a in input()/dictionary/e
where $a/hwg/hw="his"
return
    <Output>
        {$a/hwg/hw}
        {$a/hwg/pr}
        {$a/hwg/pos}
```

```
            {$a/etymology/cr[1]}
            {$a/ss/s[1]/def}
        </Output>
```

XML Collection:

```
select * from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'his'
```

SQL Server:

```
select *
from dictionary_tab,
    hw_tab,
    pr_tab,
    pos_tab,
    et_cr_tab,
    s_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_i_tab,
    qt_b_tab
where dictionary_tab.entry_id = hw_tab.entry_id and
    dictionary_tab.entry_id = pr_tab.entry_id and
    dictionary_tab.entry_id = pos_tab.entry_id and
    dictionary_tab.entry_id = et_cr_tab.entry_id and
    dictionary_tab.entry_id = s_tab.entry_id and
    dictionary_tab.entry_id = def_cr_tab.entry_id and
    dictionary_tab.entry_id = q_tab.entry_id and
    dictionary_tab.entry_id = qt_cr_tab.entry_id and
    dictionary_tab.entry_id = qt_i_tab.entry_id and
    dictionary_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.hw = 'his'
for xml auto, elements
```

- **TC/SD_Q14** *List the ids of entries that do not have variant form lists and etymologies.*

XQuery:

```
for $a in input()/dictionary/e
where empty($a/vfl) and empty($a/et)
return
    <NoVFLnET>
        {$a/@id}
    </NoVFLnET>
```

XML Collection:

```
select entry_id from dictionary_tab dict where not exists
    (select *
    from vf_tab
    where vf_tab.entry_id = dict.entry_id
    union all
    select *
    from et_cr_tab
    where et_cr_tab.entry_id = dict.entry_id)
```

SQL Server:

```
select entry_id
from dictionary_tab dict
where not exists
    (select *
    from vf_tab
    where vf_tab.entry_id = dict.entry_id
    union all
    select *
    from et_cr_tab
    where et_cr_tab.entry_id = dict.entry_id)
for xml auto
```

- **TC/SD_Q17** *Return the headwords of the entries which contain a certain word ("hockey").*

XQuery:

```
for $a in input()/dictionary/e
where contains ($a, "hockey")
return
    $a/hwg/hw
```

X-Hive/Rewrite:

```
unordered for $a in input()/dictionary/e
where xhive:fts($a, "'hockey'")
return
$a/hwg/hw
```

XML Collection:

```
select distinct(hw) from hw_tab,
    et_cr_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_b_tab,
    qt_i_tab
where hw_tab.entry_id = et_cr_tab.entry_id and
```

```
    hw_tab.entry_id = def_cr_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = qt_cr_tab.entry_id and
    hw_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.entry_id = qt_i_tab.entry_id and
    (et_cr_tab.cr like '%hockey%' or
    def_cr_tab.cr like '%hockey%' or
    q_tab.w like '%hockey%' or
    q_tab.bib like '%hockey%' or
    qt_cr_tab.cr like '%hockey%' or
    qt_b_tab.b like '%hockey%' or
    qt_i_tab.i like '%hockey%')
```

SQL Server:

```
select distinct(hw)
from hw_tab,
    et_cr_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_b_tab,
    qt_i_tab
where hw_tab.entry_id = et_cr_tab.entry_id and
    hw_tab.entry_id = def_cr_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = qt_cr_tab.entry_id and
    hw_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.entry_id = qt_i_tab.entry_id and
    (et_cr_tab.cr like '%hockey%' or
    def_cr_tab.cr like '%hockey%' or
    q_tab.w like '%hockey%' or
    q_tab.bib like '%hockey%' or
    qt_cr_tab.cr like '%hockey%' or
    qt_b_tab.b like '%hockey%' or
    qt_i_tab.i like '%hockey%')
for xml auto
```

- **TC/SD_Q18** *List the headwords of entries which contain a given phrase ("the hockey").*

  XQuery:

  ```
  for $a in input()/dictionary/e
  where contains($a, "the hockey")
  return
      $a/hwg/hw
  ```

  X-Hive/Rewrite:

  ```
  unordered for $a in input()/dictionary/e
  where xhive:fts($a, "'the hockey'")
  return
  $a/hwg/hw
  ```

  XML Collection:

  ```
  select distinct(hw) from hw_tab,
      et_cr_tab,
  ```

```
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_b_tab,
    qt_i_tab
where hw_tab.entry_id = et_cr_tab.entry_id and
    hw_tab.entry_id = def_cr_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = qt_cr_tab.entry_id and
    hw_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.entry_id = qt_i_tab.entry_id and
    (et_cr_tab.cr like '%hockey fan%' or
    def_cr_tab.cr like '%hockey fan%' or
    q_tab.w like '%hockey fan%' or
    q_tab.bib like '%hockey fan%' or
    qt_cr_tab.cr like '%hockey fan%' or
    qt_b_tab.b like '%hockey fan%' or
    qt_i_tab.i like '%hockey fan%')
```

SQL Server:

```
select distinct(hw)
from hw_tab,
    et_cr_tab,
    def_cr_tab,
    q_tab,
    qt_cr_tab,
    qt_b_tab,
    qt_i_tab
where hw_tab.entry_id = et_cr_tab.entry_id and
    hw_tab.entry_id = def_cr_tab.entry_id and
    hw_tab.entry_id = q_tab.entry_id and
    hw_tab.entry_id = qt_cr_tab.entry_id and
    hw_tab.entry_id = qt_b_tab.entry_id and
    hw_tab.entry_id = qt_i_tab.entry_id and
    (et_cr_tab.cr like '%hockey fan%' or
    def_cr_tab.cr like '%hockey fan%' or
    q_tab.w like '%hockey fan%' or
    q_tab.bib like '%hockey fan%' or
    qt_cr_tab.cr like '%hockey fan%' or
    qt_b_tab.b like '%hockey fan%' or
    qt_i_tab.i like '%hockey fan%')
for xml auto
```

- **TC/SD_Q19** Retrieve the headwords of entries cited, in etymology part, by certain entry with id attribute value (E1).

XQuery:

```
for $ent in input()/dictionary/e[@id="E1"],
    $related in input()/dictionary/e
where $ent/et/cr = $related/@id
return
    <Output>
        {$related/hwg/hw}
    </Output>
```

X-Hive/Rewrite:

```
for $ent in input()/dictionary/e[@id="E1"],
$id in $ent/et/cr,
$related in input()/dictionary/e[@id = $id]
return
<Output>
  {$related/hwg/hw}
</Output>
```

XML Collection:

```
select distinct(hw) from hw_tab,
    et_cr_tab
where hw_tab.entry_id = et_cr_tab.cr and
    et_cr_tab.entry_id = 'E1'
```

SQL Server:

```
select distinct(hw)
from hw_tab,
    et_cr_tab
where hw_tab.entry_id = et_cr_tab.cr and
    et_cr_tab.entry_id = 'E1'
for xml auto
```

A.2 Text-Centric Multiple Document

The following queries are based on an implicit (unnamed) input data set, which is a collection of article documents ("articleXXX.xml").

- **TC/MD_Q1** *Return the title of the article that has matching id attribute value (1).*

  XQuery:

  ```
  for $art in input()/article[@id="1"]
  return
      $art/prolog/title
  ```

  XML Column:

  ```
  select title
  from article_side_tab
  where article_side_tab.article_id = 1
  ```

  XML Collection:

  ```
  select title
  from article_tab
  where article_id = 1
  ```

  SQL Server:

  ```
  select title
  from article_tab
  where article_id = 1
  for xml auto
  ```

- **TC/MD_Q2** *Find the title of the article authored by (Ben Yang).*

  XQuery:

```
for $prolog in input()/article/prolog
where
    $prolog/authors/author/name="Ben Yang"
return
    $prolog/title
```

XML Column:

```
select title
from article_side_tab,
    author_side_tab
where article_side_tab.id = author_side_tab.id and
    author_side_tab.name = 'Ben Yang'
```

XML Collection:

```
select title from article_tab art,
    article_author_tab auth
where art.article_id = auth.article_id and
    name = 'Ben Yang'
```

SQL Server:

```
select title
from article_tab art,
    article_author_tab auth
where art.article_id = auth.article_id and
    name = 'Ben Yang'
for xml auto
```

- **TC/MD_Q3** *Group articles by date and calculate the total number of articles in each group.*

  XQuery:

```
for $a in distinct-values (input()/article/prolog/dateline/date)
let $b := input()/article/prolog/dateline[date=$a]
return
    <Output>
        <Date>{$a/text()}</Date>
        <NumberOfArticles>{count($b)}</NumberOfArticles>
    </Output>
```

  XML Column:

```
select date,
    count(*) number
from article_side_tab
group by date
```

  XML Collection:

```
select date,
    count(*) number
from article_tab group by date
```

  SQL Server:

```
select *
from
(select date,
    count(*) number
```

```
from article_tab
group by date) temp
for xml auto
```

- **TC/MD_Q4** *Find the heading of the section following the section entitled "Introduction" in a certain article with id attribute value (8).*

  XQuery:

```
for $a in input()/article[@id="8"]/body/section
    [@heading="introduction"],
    $p in input()/article[@id="8"]/body/section
    [. >> $a][1]
return
    <HeadingOfSection>
        {$p/@heading}
    </HeadingOfSection>
```

  XML Column:

```
select s2.heading from article_side_tab a,
    section_side_tab s1,
    section_side_tab s2
where a.id = s1.id and
    a.id = s2.id and
    a.article_id = 8 and
    s1.heading = 'introduction' and
    s1.dxx_seqno = s2.dxx_seqno - 1
```

  XML Collection:

```
select sec_heading
from article_tab art,
    body_sec_tab sec
where art.article_id = sec.article_id and
    art.article_id = '8' and
    sec_heading = 'Introduction'
```

  SQL Server:

```
select sec_heading
from article_tab art,
    body_sec_tab sec
where art.article_id = sec.article_id and
    art.article_id = '8' and
    sec_heading = 'Introduction'
for xml auto
```

- **TC/MD_Q5** *Return the headings of the first section of a certain article with id attribute value (9).*

  XQuery:

```
for $a in input()/article[@id="9"]
return
    <HeadingOfSection>
        {$a/body/section[1]/@heading}
    </HeadingOfSection>
```

  XML Column:

```
select heading
from article_side_tab a,
    section_side_tab s
where a.id = s.id and
    a.article_id = 9 and
    s.dxx_seqno = 1
```

XML Collection:

```
select sec_heading
from article_tab art,
    body_sec_tab sec
where art.article_id = sec.article_id and
    art.article_id = '9'
```

SQL Server:

```
select sec_heading
from article_tab art,
    body_sec_tab sec
where art.article_id = sec.article_id and
    art.article_id = '9'
for xml auto
```

- **TC/MD_Q6** *Find titles of articles where both keywords ("the" and "hockey") are mentioned in the same paragraph of abstracts.*

XQuery:

```
for $a in input()/article
where some $b in $a/body/abstract/p satisfies
    (contains($b, "the") and contains($b, "hockey"))
return
    $a/prolog/title
```

X-Hive/Rewrite:

```
unordered for $a in input()/article
where $a/body/abstract/p[xhive:fts(., 'the AND hockey')]
return
$a/prolog/title
```

XML Column:

```
select title
from article_side_tab a
where exists
    (select id
    from p_side_tab p
    where p.id = a.id and
    (p like '%the%hockey%' or
    p like '%hockey%the%'))
```

XML Collection:

```
select title
from article_tab art
where exists
    (select t1.article_id
    from body_p_tab t1
```

```
    where art.article_id = t1.article_id and
    (t1.p like '%the%hockey%' or
    t1.p like '%hockey%the%'))
```

SQL Server:

```
select title
from article_tab art
where exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    (t1.p like '%the%hockey%' or
    t1.p like '%hockey%the%'))
for xml auto
```

- **TC/MD_Q7** *Find titles of articles where a keyword ("hockey") is mentioned in every paragraph of abstract.*

  XQuery:

```
for $a in input()/article
where every $b in $a/body/abstract/p satisfies
    contains($b, "hockey")
return
    $a/prolog/title
```

  XML Column:

```
select title
from article_side_tab a
where not exists
    (select id
    from p_side_tab p
    where p.id = a.id and
    p not like '%hockey%')
```

  XML Collection:

```
select title
from article_tab art
where not exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p not like '%hockey%')
```

  SQL Server:

```
select title
from article_tab art
where not exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p not like '%hockey%')
for xml auto
```

- **TC/MD_Q8** *Return the names of all authors (one element name unknown) of the article with matching id attribute value (2).*

XQuery:

```
for $art in input()/article[@id="2"]
return
    $art/prolog/*/author/name
```

XML Column:

```
select name
from article_side_tab art,
    author_side_tab auth
where art.id = auth.id and
    art.article_id = 2
```

XML Collection:

```
select name
from article_author_tab
where article_id = 2
```

SQL Server:

```
select name
from article_author_tab
where article_id = 2
for xml auto
```

- **TC/MD_Q9** *Return all author names (several consecutive element unknown) of the article with matching id attribute value (3).*

  XQuery:

  ```
  for $art in input()/article[@id="3"]
  return
      $art//author/name
  ```

  XML Column:

  ```
  select name
  from article_side_tab art,
      author_side_tab auth
  where art.id = auth.id and
      art.article_id = 3
  ```

  XML Collection:

  ```
  select name
  from article_author_tab
  where article_id = 3
  ```

  SQL Server:

  ```
  select name
  from article_author_tab
  where article_id = 3
  for xml auto
  ```

- **TC/MD_Q10** *List the titles of articles sorted by country.*

  XQuery:

```
for $a in input()/article/prolog
order by $a/dateline/country
return
    <Output>
        {$a/title}
        {$a/dateline/country}
    </Output>
```

X-Hive:

```
for $a in input()/article/prolog
return
    <Output>
        {$a/title}
        {$a/dateline/country}
    </Output>
sort by (country)
```

XML Column:

```
select title, country
from article_side_tab
order by country
```

XML Collection:

```
select title, country
from article_tab
order by country
```

SQL Server:

```
select title, country
from article_tab
order by country
for xml auto, elements
```

- **TC/MD_Q11** *List the titles of articles that have a matching country element type (Canada), sorted by date.*

  XQuery:

```
for $a in input()/article/prolog
where $a/dateline/country="Canada"
order by $a/dateline/date
return
    <Output>
        {$a/title}
        {$a/dateline/date}
    </Output>
```

  X-Hive:

```
for $a in input()/article/prolog
where $a/dateline/country="Canada"
return
    <Output>
        {$a/title}
        {$a/dateline/date}
    </Output>
sort by (date)
```

XML Column:

```
select title, date
from article_side_tab
where country = 'Canada'
order by date
```

XML Collection:

```
select title, date
from article_tab
where country = 'Canada'
order by date
```

SQL Server:

```
select title, date
from article_tab
where country = 'Canada'
order by date
for xml auto, elements
```

- **TC/MD_Q12** *Retrieve the body of the article that has a matching id attribute value (4).*

XQuery:

```
for $a in input()/article[@id="4"]
return
    <Article>
        {$a/body}
    </Article>
```

XML Column:

```
select db2xml.extractClob(article, '/article/body') body
from article_tab a,
    article_side_tab s
where a.id = s.id and
    s.article_id = 4
```

XML Collection:

```
select *
from article_tab,
    body_p_tab,
    body_sec_tab,
    sec_p_tab,
    sec_subsec_tab,
    subsec_p_tab,
    subsec_subsubsec_tab,
    subsubsec_p_tab,
    subsubsec_subsubsubsec_tab,
    subsubsubsec_p_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = body_sec_tab.article_id and
    article_tab.article_id = sec_p_tab.article_id and
    article_tab.article_id = sec_subsec_tab.article_id and
    article_tab.article_id = subsec_p_tab.article_id and
    article_tab.article_id = subsec_subsubsec_tab.article_id and
    article_tab.article_id = subsubsec_p_tab.article_id and
```

```
    article_tab.article_id =
        subsubsec_subsubsubsec_tab.article_id and
    article_tab.article_id = subsubsubsec_p_tab.article_id and
    article_tab.article_id = 4
```

SQL Server:

```
select *
from article_tab,
    body_p_tab,
    body_sec_tab,
    sec_p_tab,
    sec_subsec_tab,
    subsec_p_tab,
    subsec_subsubsec_tab,
    subsubsec_p_tab,
    subsubsec_subsubsubsec_tab,
    subsubsubsec_p_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = body_sec_tab.article_id and
    article_tab.article_id = sec_p_tab.article_id and
    article_tab.article_id = sec_subsec_tab.article_id and
    article_tab.article_id = subsec_p_tab.article_id and
    article_tab.article_id = subsec_subsubsec_tab.article_id and
    article_tab.article_id = subsubsec_p_tab.article_id and
    article_tab.article_id =
        subsubsec_subsubsubsec_tab.article_id and
    article_tab.article_id = subsubsubsec_p_tab.article_id and
    article_tab.article_id = 4
for xml auto, elements
```

- **TC/MD_Q13** *Construct a brief information on the article that has a matching id attribute value (5), including title, the name of first author, date and abstract.*

XQuery:

```
for $a in input()/article[@id="5"]
return
    <Output>
        {$a/prolog/title}
        {$a/prolog/authors/author[1]/name}
        {$a/prolog/dateline/date}
        {$a/body/abstract}
    </Output>
```

XML Column:

```
select title,
    name,
    date,
    db2xml.extractClob(article, '/article/body/abstract') abs
from article_tab a,
    article_side_tab art,
    author_side_tab auth
where a.id = art.id and
    art.id = auth.id and
    art.article_id = 5 and
    auth.dxx_seqno = 1
```

XML Collection:

```
select title,
    name,
    date,
    p
from article_tab,
    body_p_tab,
    article_author_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = article_author_tab.article_id and
    article_tab.article_id = 5
```

SQL Server:

```
select title,
    name,
    date,
    p
from article_tab,
    body_p_tab,
    article_author_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = article_author_tab.article_id and
    article_tab.article_id = 5
for xml auto, elements
```

- **TC/MD_Q14** *List article title that doesn't have genre element.*

  XQuery:

  ```
  for $a in input()/article/prolog
  where empty ($a/genre)
  return
      <NoGenre>
          {$a/title}
      </NoGenre>
  ```

  XML Column:

  ```
  select title
  from article_side_tab
  where genre is null
  ```

  XML Collection:

  ```
  select title
  from article_tab
  where genre is null
  ```

  SQL Server:

  ```
  select title
  from article_tab
  where genre is null
  for xml auto
  ```

- **TC/MD_Q15** *List author names whose contact elements are empty in articles.*

  XQuery:

```
for $a in input()/article/prolog/authors/author
where empty($a/contact/text())
return
    <NoContact>
        {$a/name}
    </NoContact>
```

XML Column:

```
select name
from author_side_tab a
where not exists
    (select *
    from phone_side_tab p
    where a.id = p.id)
    and not exists
    (select *
    from email_side_tab e
    where a.id = e.id)
```

XML Collection:

```
select name
from article_tab,
    article_author_tab
where article_tab.article_id = article_author_tab.article_id and
    phone is null and
    email is null
```

SQL Server:

```
select name
from article_tab,
    article_author_tab
where article_tab.article_id = article_author_tab.article_id and
    phone is null and
    email is null
for xml auto
```

- **TC/MD_Q16** *Get the article by its id attribute value (6).*

XQuery:

```
for $a in input()/article[@id="6"]
return
    $a
```

XML Column:

```
select db2xml.clob(article)
 from article_tab a,
     article_side_tab s
 where a.id = s.id and
     s.article_id = 6
```

XML Collection:

```
select * from article_tab,
    article_author_tab,
    keyword_tab,
```

```
    body_p_tab,
    body_sec_tab,
    sec_p_tab,
    sec_subsec_tab,
    subsec_p_tab,
    subsec_subsubsec_tab,
    subsubsec_p_tab,
    subsubsec_subsubsubsec_tab,
    subsubsubsec_p_tab,
    ack_tab,
    ref_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = body_sec_tab.article_id and
    article_tab.article_id = sec_p_tab.article_id and
    article_tab.article_id = sec_subsec_tab.article_id and
    article_tab.article_id = subsec_p_tab.article_id and
    article_tab.article_id = subsec_subsubsec_tab.article_id and
    article_tab.article_id = subsubsec_p_tab.article_id and
    article_tab.article_id =
        subsubsec_subsubsubsec_tab.article_id and
    article_tab.article_id = subsubsubsec_p_tab.article_id and
    article_tab.article_id = article_author_tab.article_id and
    article_tab.article_id = keyword_tab.article_id and
    article_tab.article_id = ack_tab.article_id and
    article_tab.article_id = ref_tab.article_id and
    article_tab.article_id = 6
```

SQL Server:

```
select *
from article_tab,
    article_author_tab,
    keyword_tab,
    body_p_tab,
    body_sec_tab,
    sec_p_tab,
    sec_subsec_tab,
    subsec_p_tab,
    subsec_subsubsec_tab,
    subsubsec_p_tab,
    subsubsec_subsubsubsec_tab,
    subsubsubsec_p_tab,
    ack_tab,
    ref_tab
where article_tab.article_id = body_p_tab.article_id and
    article_tab.article_id = body_sec_tab.article_id and
    article_tab.article_id = sec_p_tab.article_id and
    article_tab.article_id = sec_subsec_tab.article_id and
    article_tab.article_id = subsec_p_tab.article_id and
    article_tab.article_id = subsec_subsubsec_tab.article_id and
    article_tab.article_id = subsubsec_p_tab.article_id and
    article_tab.article_id =
        subsubsec_subsubsubsec_tab.article_id and
    article_tab.article_id = subsubsubsec_p_tab.article_id and
    article_tab.article_id = article_author_tab.article_id and
```

```
        article_tab.article_id = keyword_tab.article_id and
        article_tab.article_id = ack_tab.article_id and
        article_tab.article_id = ref_tab.article_id and
        article_tab.article_id = 6
    for xml auto, elements
```

- **TC/MD_Q17** *Return the titles of articles which contain a certain word ("hockey").*

  XQuery:

```
for $a in input()/article
where contains ($a//p, "hockey")
return
    $a/prolog/title
```

  XML Column:

```
select distinct(title)
from article_side_tab a
where exists
    (select *
    from p_side_tab p
    where a.id = p.id and
    p.p like '%hockey%')
```

  XML Collection:

```
select distinct(title)
from article_tab art
where exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from sec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsubsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%')
```

  SQL Server:

```
select distinct(title)
from article_tab art
where exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from sec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%'
    union all
    select t1.article_id
    from subsubsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%hockey%')
for xml auto
```

- **TC/MD_Q18** *List the titles and abstracts of articles which contain a given phrase ("the hockey").*

  XQuery:

```
for $a in input()/article
where contains ($a//p, "the hockey")
return
    <Output>
        {$a/prolog/title}
        {$a/body/abstract}
    </Output>
```

  XML Column:

```
select title,
    db2xml.extractClob(article, '/article/body/abstract') abs
from    article_tab a,
    article_side_tab s
where a.id = s.id and
    exists
    (select *
    from p_side_tab p
    where s.id = p.id and
    p.p like '%the hockey%')
```

  XML Collection:

```
select distinct(title),
    p
from article_tab art,
    body_p_tab p
where art.article_id = p.p and
    exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from sec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from subsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from subsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from subsubsubsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%')
```

SQL Server:

```
select distinct(title),
    p
from article_tab art,
    body_p_tab p
where art.article_id = p.p and
    exists
    (select t1.article_id
    from body_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from sec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
    from subsec_p_tab t1
    where art.article_id = t1.article_id and
    t1.p like '%the hockey%'
    union all
    select t1.article_id
```

```
        from subsubsec_p_tab t1
        where art.article_id = t1.article_id and
        t1.p like '%the hockey%'
        union all
        select t1.article_id
        from subsubsubsec_p_tab t1
        where art.article_id = t1.article_id and
        t1.p like '%the hockey%')
    for xml auto
```

- **TC/MD_Q19** *List the names of articles cited by an article with a certain id attribute value (7).*

  XQuery:

```
for $a in input()/article[@id='7']/epilog/references/a_id,
    $b in input()/article
where $a = $b/@id
return
    <Output>
        {$b/prolog/title}
    </Output>
```

  XML Column:

```
select a2.title
from article_side_tab a1,
    ref_side_tab r,
    article_side_tab a2
where a1.id = 7 and
    r.id = a1.id and
    r.a_id = a2.article_id
```

  XML Collection:

```
select title
from article_tab art,
    ref_tab ref
where ref.article_id = 7 and
    ref.a_id = art.article_id
```

  SQL Server:

```
select title
from article_tab art,
    ref_tab ref
where ref.article_id = 7 and
    ref.a_id = art.article_id
for xml auto
```

A.3 Data-Centric Single Document

The following queries are based on an implicit (unnamed) input data set, which is "catalog.xml".

- **DC/SD_Q1** *Return the item that has matching item id attribute value (I1).*

  XQuery:

```
for $item in input()/catalog/:item[@id="I1"]
return
    $item
```

XML Collection:

```
select * from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = 'I1' and
    cat.item_id = auth.item_id and
    cat.item_id = auth_add.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
```

SQL Server:

```
select *
from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = 'I1' and
    cat.item_id = auth.item_id and
    cat.item_id = auth_add.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
for xml auto, elements
```

- **DC/SD_Q2** *Find the title of the item which has matching author first name (Ben).*

XQuery:

```
for $item in input()/catalog/:item
where $item/authors/author/name/first_name = "Ben"
return
    $item/title
```

X-Hive/Rewrite:

```
unordered for $item in input()/catalog/:item
where $item/authors/author/name/first_name = "Ben"
return
    $item/title
```

XML Collection:

```
select title
from catalog_tab,catalog_author_tab
where catalog_tab.item_id = catalog_author_tab.item_id and
    first_name = 'Ben'
```

SQL Server:

```
select title
from catalog_tab,catalog_author_tab
where catalog_tab.item_id = catalog_author_tab.item_id and
    first_name = 'Ben'
for xml auto
```

- **DC/SD_Q3** *Group items released in a certain year (1990), by publisher name and calculate the total number of items for each group.*

XQuery:

```
for $a in distinct-values (input()/catalog/:item
    [date_of_release >= "1990-01-01"]
    [date_of_release < "1991-01-01"]/publisher/name)
let $b := input()/catalog/:item/publisher[name=$a]
return
    <Output>
        <Publisher>{$a/text()}</Publisher>
        <NumberOfItems>{count($b)}</NumberOfItems>
    </Output>
```

X-Hive/Rewrite:

```
unordered for $a in distinct-values (input()/catalog/:item
[date_of_release[. >= "1990-01-01" and . < "1991-01-01"]]/publisher/name)
let $b := unordered input()/catalog/:item/publisher[xhive:fts(name,
    concat("'", $a, "'"))]
return
<Output>
  <Publisher>
    {$a/text()}
  </Publisher>
  <NumberOfItems>
    {count($b)}
  </NumberOfItems>
</Output>
```

XML Collection:

```
select publisher_name,
    count(*) number_of_items
from catalog_tab
where date_of_release >= '1990-01-01' and
    date_of_release < '1990-01-01'
group by publisher_name
```

SQL Server:

```
select *
from
(select publisher_name,
    count(*) number_of_items
from catalog_tab
where date_of_release >= '1990-01-01' and
    date_of_release < '1990-01-01'
group by publisher_name) temp
for xml auto, elements
```

- **DC/SD_Q4** *List the item id of the previous item of a matching item with id attribute value (I2).*

  XQuery:

```
let $item := input()/catalog/:item[@id="I2"]
for  $prevItem in input()/catalog/:item
    [. << $item][position() = last()]
return
    <Output>
        <CurrentItem>{$item/@id}</CurrentItem>
        <PreviousItem>{$prevItem/@id}</PreviousItem>
    </Output>
```

  X-Hive/Rewrite:

```
let $item := input()/catalog/:item[@id="I2"]
for $prevItem in $item/preceding:::item[1]
return
<Output>
  <CurrentItem>
    {$item/@id}
  </CurrentItem>
  <PreviousItem>
    {$prevItem/@id}
  </PreviousItem>
</Output>
```

  XML Collection:

```
select t1.item_id curr,
    max(t2.item_id) prev
from catalog_tab t1, catalog_tab t2
where t1.item_id = 'I2' and
    t1.item_id > t2.item_id
group by t1.item_id
```

  SQL Server:

```
select *
from
(select t1.item_id curr,
    max(t2.item_id) prev
from catalog_tab t1, catalog_tab t2
where t1.item_id = 'I2' and
    t1.item_id > t2.item_id
group by t1.item_id) temp
for xml auto, elements
```

- **DC/SD_Q5** *Return the information about the first author of item with a matching id attribute value (I3).*

  XQuery:

```
for $a in input()/catalog/:item[@id="I3"]
return
    $a/authors/author[1]
```

  XML Collection:

```
SELECT *
FROM catalog_author_tab auth,
    catalog_author_address_tab auth_add
WHERE auth.item_id = 'I3' AND
    auth.item_id = auth_add.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
```

SQL Server:

```
SELECT *
FROM catalog_author_tab auth,
    catalog_author_address_tab auth_add
WHERE auth.item_id = 'I3' AND
    auth.item_id = auth_add.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
FOR XML AUTO, ELEMENTS
```

- **DC/SD_Q6** *Return item information where some authors are from certain country (Canada).*

  XQuery:

```
for $item in input()/catalog/:item
where some $auth in
    $item/authors/author/contact_information/mailing_address
satisfies $auth/name_of_country = "Canada"
return
    $item
```

  X-Hive/Rewrite:

```
unordered for $item in input()/catalog/:item
where
    $item/authors/author/contact_information/mailing_address/
    name_of_country[. = "Canada"]
return
$item
```

  XML Collection:

```
select *
from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = auth.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name and
    exists
    (select t1.item_id
    from catalog_author_tab t1
```

```
        where t1.item_id = cat.item_id and
            name_of_country = 'Canada')
```

SQL Server:

```
select *
from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = auth.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name and
    exists
    (select t1.item_id
    from catalog_author_tab t1
    where t1.item_id = cat.item_id and
        name_of_country = 'Canada')
for xml auto, elements
```

- **DC/SD_Q7** *Return item information where all its authors are from certain country (Canada).*

  XQuery:

```
for $item in input()/catalog/:item
where every $add in
    $item/authors/author/contact_information/mailing_address
satisfies $add/name_of_country = "Canada"
return
    $item
```

  XML Collection:

```
select *
from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = auth.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name and
    not exists
    (select t1.item_id
    from catalog_author_tab t1
    where t1.item_id = cat.item_id and
        name_of_country <> 'Canada')
```

  SQL Server:

```
select *
from catalog_tab cat,
    catalog_author_tab auth,
    catalog_author_address_tab auth_add,
    catalog_publisher_address_tab pub_add,
    catalog_related_item_tab rel
where   cat.item_id = auth.item_id and
    cat.item_id = pub_add.item_id and
    cat.item_id = rel.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name and
    not exists
    (select t1.item_id
    from catalog_author_tab t1
    where t1.item_id = cat.item_id and
        name_of_country <> 'Canada')
for xml auto, elements
```

- **DC/SD_Q8** *Return the publisher of an item with id attribute value (I4).*

  XQuery:

```
for $a in input()/catalog/*[@id="I4"]
return
    $a/publisher
```

  XML Collection:

```
select
    publisher_name,
    publisher_name_of_city,
    publisher_name_of_state,
    publisher_zip_code,
    publisher_country_name,
    publisher_country_ex_rate,
    publisher_country_currency,
    publisher_FAX_number,
    publisher_phone_number,
    publisher_web_site,
    street_address
from catalog_tab c,
    catalog_publisher_address_tab p
where c.item_id = p.item_id and
    c.item_id = 'I4'
```

  SQL Server:

```
select 1 tag,
  null parent,
  publisher_name as [publisher!1!name!element],
  publisher_name_of_city as [publisher!1!name_of_city!element],
  publisher_name_of_state as [publisher!1!name_of_state!element],
  publisher_zip_code as [publisher!1!zip_code!element],
  publisher_country_name as [publisher!1!country_name!element],
  publisher_country_exchange_rate as
    [publisher!1!exchange_rate!element],
```

```
  publisher_country_currency as [publisher!1!currency!element],
  publisher_FAX_number as [publisher!1!FAX!element],
  publisher_phone_number as [publisher!1!phone!element],
  publisher_web_site as [publisher!1!web_site!element],
  null as [street_info!2!street_address!element]
from catalog_tab
where item_id = 'I4'
union all
select 2,
    1,
    NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
    street_address
from catalog_publisher_address_tab
where item_id = 'I4'
for xml explicit
```

- **DC/SD_Q9** *Return the ISBN of an item with id attribute value (I5).*

  XQuery:

```
for $a in input()/catalog/:item
where $a/@id="I5"
return
    $a//ISBN/text()
```

  XML Collection:

```
select ISBN
from catalog_tab
where item_id = 'I5'
```

  SQL Server:

```
select ISBN
from catalog_tab
where item_id = 'I5'
for xml auto
```

- **DC/SD_Q10** *List the item titles ordered alphabetically by publisher name, with release date within a certain time period (from 1990-01-01 to 1995-01-01).*

  XQuery:

```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
    $a/date_of_release lt "1995-01-01"
order by $a/publisher/name
return
    <Output>
```

```
            {$a/title}
            {$a/publisher}
        </Output>
```

X-Hive:

```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
    $a/date_of_release lt "1995-01-01"
return
    <Output>
        {$a/title}
        {$a/publisher}
    </Output>
sort by (publisher/name)
```

X-Hive/Rewrite:

```
for $a in input()/catalog/:item
where $a/date_of_release[. gt "1990-01-01" and . lt "1995-01-01"]
return
<Output>
  {$a/title}
  {$a/publisher}
</Output>
sort by (publisher/name)
```

XML Collection:

```
select title, publisher_name
from catalog_tab
where date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
order by publisher_name
```

SQL Server:

```
select title, publisher_name
from catalog_tab
where date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
order by publisher_name
for xml auto, elements
```

- **DC/SD_Q11** *List the item titles in descending order by date of release with date of release within a certain time range (from 1990-01-01 to 1995-01-01.*

XQuery:

```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
    $a/date_of_release lt "1995-01-01"
order by $a/data_of_release descending
return
    <Output>
        {$a/title}
        {$a/date_of_release}
    </Output>
```

X-Hive:

```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
    $a/date_of_release lt "1995-01-01"
return
    <Output>
        {$a/title}
        {$a/date_of_release}
    </Output>
sort by ((date_of_realse) descending)
```

X-Hive/Rewrite:

```
for $a in input()/catalog/:item
where $a/date_of_release[. gt "1990-01-01" and . lt "1995-01-01"]
return
<Output>
  {$a/title}
  {$a/date_of_release}
</Output>
sort by ((date_of_release) descending)
```

XML Collection:

```
select title, date_of_release
from catalog_tab
where date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
order by date_of_release desc
```

SQL Server:

```
select title, date_of_release
from catalog_tab
where date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
order by date_of_release desc
for xml auto, elements
```

- **DC/SD_Q12** *Get the mailing address of the first author of certain item with id attribute value (I6).*

XQuery:

```
for $a in input()/catalog/:item[@id="I6"]
return
    <Output>
        {$a/authors/author[1]/contact_information/mailing_address}
    </Output>
```

XML Collection:

```
select *
from catalog_author_tab auth,
    catalog_author_address_tab auth_add
where   auth.item_id = 'I6' and
    auth.item_id = auth_add.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
```

```
order by auth.item_id,
    auth.first_name,
    auth.middle_name,
    auth.last_name
```

SQL Server:

```
select *
from catalog_author_tab auth,
    catalog_author_address_tab auth_add
where   auth.item_id = 'I6' and
    auth.item_id = auth_add.item_id and
    auth.first_name = auth_add.first_name and
    auth.middle_name = auth_add.middle_name and
    auth.last_name = auth_add.last_name
order by auth.item_id,
    auth.first_name,
    auth.middle_name,
    auth.last_name
for xml auto, elements
```

- **DC/SD_Q14** *Return the names of publishers who publish books between a period of time (from 1990-01-01 to 1991-01-01) but do not have FAX number.*

XQuery:

```
for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
           $a/date_of_release lt "1991-01-01" and
           empty($a/publisher/contact_information/FAX_number)
return
      <Output>
        {$a/publisher/name}
      </Output>
```

X-Hive/Rewrite:

```
unordered for $a in input()/catalog/:item
where $a/date_of_release[. gt "1990-01-01" and . lt "1991-01-01"] and
empty($a/publisher/contact_information/FAX_number)
return
<Output>
  {$a/publisher/name}
</Output>

or even faster:

let $item := (unordered for $a in input()/catalog/:item
where $a/date_of_release[. gt "1990-01-01" and . lt "1991-01-01"]
return $a)
for $i in unordered $item
where empty($i/publisher/contact_information/FAX_number)
return
<Output>
  {$i/publisher/name}
</Output>
```

XML Collection:

```
select distinct(publisher_name)
from catalog_tab
where publisher_FAX_number is null and
    date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
```

SQL Server:

```
select distinct(publisher_name)
from catalog_tab
where publisher_FAX_number = null and
    date_of_release > '1990-01-01' and
    date_of_release < '1995-01-01'
for xml auto
```

- **DC/SD_Q17** *Return the ids of items whose descriptions contain a certain word ("hockey").*

  XQuery:

```
for $a in input()/catalog/:item
where contains ($a/description, "hockey")
return
<Output>
    {$a/@id}
</Output>
```

  X-Hive/Rewrite:

```
unordered for $a in input()/catalog/:item
where xhive:fts($a/description, "hockey")
return
<Output>
  {$a/@id}
</Output>
```

  XML Collection:

```
select item_id
from catalog_tab
where catalog_tab.description like '%hockey%'
```

  SQL Server:

```
select item_id
from catalog_tab
where catalog_tab.description like '%hockey%'
for xml auto
```

- **DC/SD_Q19** *Retrieve the item titles related by certain item with id attribute value (I7).*

  XQuery:

```
for $item in input()/catalog/:item[@id="I7"],
    $related in input()/catalog/:item
where $item/related_items/related_item/item_id = $related/@id
return
    <Output>
        {$related/title}
    </Output>
```

  X-Hive/Rewrite:

```
for $item in input()/catalog/:item[@id="I7"],
$id in $item/related_items/related_item/item_id,
$related in input()/catalog/:item[@id = $id]
return
<Output>
  {$related/title}
</Output>
```

XML Collection:

```
select title
from catalog_tab cat,
    catalog_related_item_tab rel
where rel.item_id = 'I7' and
    rel.related_item_id = cat.item_id
```

SQL Server:

```
select title
from catalog_tab cat,
    catalog_related_item_tab rel
where rel.item_id = 'I7' and
    rel.related_item_id = cat.item_id
for xml auto
```

- **DC/SD_Q20** *Retrieve the item title whose size (length*width*height) is bigger than certain number (500000).*

  XQuery:

```
for $size in input()/catalog/:item/attributes/size_of_book
where $size/length*$size/width*$size/height > 500000
return
    <Output>
        {$size/../../title}
    </Output>
```

  XML Collection:

```
select title
from catalog_tab
where length * width * height > 500000
```

  SQL Server:

```
select title
from catalog_tab
where length * width * height > 500000
for xml auto
```

A.4 Data-Centric Multiple Document

The following queries are based on an implicit (unnamed) input data set, which is a collection of documents ("orderXXX.xml", "customer.xml", "item.xml", "author.xml", "address.xml" and "country.xml").

- **DC/MD_Q1** *Return the customer id of the order that has matching id attribute value (1).*

  XQuery:

```
for $order in input()/order[@id="1"]
return
    $order/customer_id
```

XML Column:

```
SELECT customer_id
FROM order_side_tab
WHERE order_id = 1
```

XML Collection:

```
SELECT customer_id
FROM order_tab
WHERE order_id = 1
```

SQL Server:

```
SELECT customer_id
FROM order_tab
WHERE order_id = 1
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q3** *Group orders with total amount bigger than a certain number (11000.0), by customer id and calculate the total number of each group.*

XQuery:

```
for $a in distinct-values (input()/order
    [total > 11000.0]/customer_id)
let $b := input()/order[customer_id=$a]
return
    <Output>
        <CustKey>{$a/text()}</CustKey>
        <NumberOfOrders>{count($b)}</NumberOfOrders>
    </Output>
```

X-Hive/Rewrite:

```
unordered for $a in distinct-values (input()/order
[total > double(11000.0)]/customer_id)
let $b := unordered input()/order[customer_id=$a]
return
<Output>
  <CustKey>
    {$a/text()}
  </CustKey>
  <NumberOfOrders>
    {count($b)}
  </NumberOfOrders>
</Output>
```

XML Column:

```
SELECT customer_id,
    COUNT(*) number
FROM order_side_tab
WHERE total > 11000.0
GROUP BY customer_id
```

XML Collection:

```
SELECT customer_id,
    COUNT(*) number
FROM order_tab
WHERE total > 11000.0
GROUP BY customer_id
```

SQL Server:

```
SELECT * FROM
(SELECT customer_id,
    COUNT(*) number
FROM order_tab
WHERE total > 11000.0
GROUP BY customer_id) OUTPUT
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q4** *List the item id of the previous item of a matching item with id attribute value (8).*

XQuery:

```
let $item := input()/items/:item[@id="8"],
for $prevItem in input()/items/:item
    [. << $item][position() = last()]
return
    <Output>
        <CurrentItem>{$item/@id}</CurrentItem>
        <PreviousItem>{$prevItem/@id}</PreviousItem>
    </Output>
```

X-Hive/Rewrite:

```
let $item := input()/items/:item[@id="8"]
for $prevItem in $item/preceding:::item[1]
return
<Output>
  <CurrentItem>
    {$item/@id}
  </CurrentItem>
  <PreviousItem>
    {$prevItem/@id}
  </PreviousItem>
</Output>
```

XML Column:

```
SELECT i2.item_id
FROM item_side_tab i1,
    item_side_tab i2
WHERE i1.item_id = 8 AND
    i1.dxx_seqno = i2.dxx_seqno + 1
```

XML Collection:

```
SELECT t1.order_id,
    MAX(t2.order_id)
FROM order_tab t1,order_tab t2
WHERE t1.order_id = 8 AND
    t1.order_id > t2.order_id
GROUP BY t1.order_id
```

SQL Server:

```
SELECT 1 tag,
    NULL AS parent,
    t1.order_id AS [output!1!current_order!element],
    MAX(t2.order_id) AS [output!1!previous_order!element]
FROM order_tab t1,order_tab t2
WHERE t1.order_id = 8 AND
    t1.order_id > t2.order_id
GROUP BY t1.order_id
FOR XML EXPLICIT
```

- **DC/MD_Q5** *Return the first order line item of a certain order with id attribute value (2).*

  XQuery:

  ```
  for $a in input()/order[@id="2"]
  return
      $a/order_lines/order_line[1]
  ```

  XML Column:

  ```
  select order_line_id,
      item_id,
      quantity_of_item,
      discount_rate,
      special_instructions
  from order_side_tab order,
      order_line_side_tab line,
      order_line_item_side_tab item,
      order_line_quan_side_tab quan,
      order_line_disc_side_tab disc,
      order_line_instr_side_tab instr,
      (select min(dxx_seqno) as first_id
      from  order_side_tab order,
      order_line_side_tab line
      where order.id = line.id and
          order.order_id = 2) temp
  where order.id = line.id and
      order.id = item.id and
      order.id = quan.id and
      order.id = disc.id and
      order.id = instr.id and
      order.order_id = 2 and
      line.dxx_seqno = temp.first_id
  ```

  XML Collection:

  ```
  SELECT *
  FROM order_line_tab,
  (SELECT MIN(order_line_id) AS first_id
  FROM order_line_tab
  WHERE order_id = 2) temp
  WHERE order_id = 2 AND
      order_line_id = temp.first_id
  ```

  SQL Server:

```
SELECT *
FROM order_line_tab,
(SELECT MIN(order_line_id) AS first_id
FROM order_line_tab
WHERE order_id = 2) temp
WHERE order_id = 2 AND
    order_line_id = temp.first_id
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q6** *Return invoice where some discount rates of sub-line items are higher than a certain number (0.02).*

  XQuery:

```
for $ord in input()/order
where some $item in $ord/order_lines/order_line
    satisfies $item/discount_rate gt 0.02
return
    $ord
```

  X-Hive/Rewrite:

```
unordered for $ord in input()/order
where $ord/order_lines/order_line/discount_rate[. gt double(0.02)]
return
$ord
```

  XML Column:

```
select db2xml.clob(order) order
from order_tab order
where exists
    (select *
    from order_line_disc_side_tab disc
    where order.id = disc.id and
    disc.discount_rate > 0.02)
```

  XML Collection:

```
SELECT *
FROM order_tab,
    order_line_tab
WHERE order_tab.order_id = order_line_tab.order_id AND
    EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate > 0.02)
```

  SQL Server:

```
SELECT 1 tag,
  NULL AS parent,
  order_tab.order_id AS [order!1!id],
  customer_id AS [order!1!customer_id!element],
  order_date AS [order!1!order_date!element],
  subtotal AS [order!1!subtotal!element],
  tax AS [order!1!tax!element],
  total AS [order!1!total!element],
```

```
  ship_type AS [order!1!ship_type!element],
  ship_date AS [order!1!ship_date!element],
  bill_address_id AS [order!1!bill_address_id!element],
  ship_address_id AS [order!1!ship_address_id!element],
  order_status AS [order!1!order_status!element],
  NULL AS [credit_card_transaction!2!credit_card_type!element],
  NULL AS [credit_card_transaction!2!credit_card_number!element],
  NULL AS [credit_card_transaction!2!name_on_credit_card!element],
  NULL AS [credit_card_transaction!2!expiration_date!element],
  NULL AS [credit_card_transaction!2!authorization_id!element],
  NULL AS [credit_card_transaction!2!transaction_amount!element],
  NULL AS [credit_card_transaction!2!authorization_date!element],
  NULL AS [credit_card_transaction!2!transaction_country_id!element],
  NULL AS [order_line!3!id],
  NULL AS [order_line!3!item_id!element],
  NULL AS [order_line!3!quantity_of_item!element],
  NULL AS [order_line!3!discount_rate!element],
  NULL AS [order_line!3!special_instructions!element]
FROM order_tab
WHERE EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate > 0.02)
UNION ALL
SELECT 2,
        1,
        order_id,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        credit_card_type,
    credit_card_number,
    name_on_credit_card,
    expiration_date,
    authorization_id,
    transaction_amount,
    authorization_date,
    transaction_country_id,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL
FROM order_tab
WHERE EXISTS
    (SELECT t1.order_id
```

```
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate > 0.02)
UNION ALL
SELECT 3,
        1,
        order_tab.order_id,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    order_line_id,
    item_id,
        quantity_of_item,
        discount_rate,
        special_instructions
FROM order_tab, order_line_tab
WHERE order_tab.order_id = order_line_tab.order_id AND
    EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate > 0.02)
ORDER BY [order!1!id],tag
FOR XML EXPLICIT
```

- **DC/MD_Q7** *Return invoice where all discount rates of sub-line items are higher than a certain number (0.02).*

  XQuery:

```
for $ord in input()/order
where every $item in $ord/order_lines/order_line
    satisfies $item/discount_rate gt 0.02
return
    $ord
```

  XML Column:

```
select db2xml.clob(order) order
from order_tab order
where not exists
```

```
    (select *
    from order_line_disc_side_tab disc
    where order.id = disc.id and
    disc.discount_rate <= 0.02)
```

XML Collection:

```
SELECT *
FROM order_tab,
    order_line_tab
WHERE order_tab.order_id = order_line_tab.order_id AND
    NOT EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate <= 0.02)
```

SQL Server:

```
SELECT 1 tag,
  NULL AS parent,
  order_tab.order_id AS [order!1!id],
  customer_id AS [order!1!customer_id!element],
  order_date AS [order!1!order_date!element],
  subtotal AS [order!1!subtotal!element],
  tax AS [order!1!tax!element],
  total AS [order!1!total!element],
  ship_type AS [order!1!ship_type!element],
  ship_date AS [order!1!ship_date!element],
  bill_address_id AS [order!1!bill_address_id!element],
  ship_address_id AS [order!1!ship_address_id!element],
  order_status AS [order!1!order_status!element],
  NULL AS [credit_card_transaction!2!credit_card_type!element],
  NULL AS [credit_card_transaction!2!credit_card_number!element],
  NULL AS [credit_card_transaction!2!name_on_credit_card!element],
  NULL AS [credit_card_transaction!2!expiration_date!element],
  NULL AS [credit_card_transaction!2!authorization_id!element],
  NULL AS [credit_card_transaction!2!transaction_amount!element],
  NULL AS [credit_card_transaction!2!authorization_date!element],
  NULL AS [credit_card_transaction!2!transaction_country_id!element],
  NULL AS [order_line!3!id],
  NULL AS [order_line!3!item_id!element],
  NULL AS [order_line!3!quantity_of_item!element],
  NULL AS [order_line!3!discount_rate!element],
  NULL AS [order_line!3!special_instructions!element]
FROM order_tab
WHERE NOT EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate <= 0.02)
UNION ALL
SELECT 2,
        1,
        order_id,
        NULL,
```

```
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        credit_card_type,
    credit_card_number,
    name_on_credit_card,
    expiration_date,
    authorization_id,
    transaction_amount,
    authorization_date,
    transaction_country_id,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL
FROM order_tab
WHERE NOT EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate <= 0.02)
UNION ALL
SELECT 3,
        1,
        order_tab.order_id,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    order_line_id,
    item_id,
        quantity_of_item,
        discount_rate,
```

```
        special_instructions
FROM order_tab, order_line_tab
WHERE order_tab.order_id = order_line_tab.order_id AND
    NOT EXISTS
    (SELECT t1.order_id
    FROM order_line_tab t1
    WHERE t1.order_id = order_tab.order_id AND
    discount_rate <= 0.02)
ORDER BY [order!1!id],tag
FOR XML EXPLICIT
```

- **DC/MD_Q8** *Return the order line item ids of an order with an attribute value (3).*

  XQuery:

  ```
  for $a in input()/order[@id="3"]
  return
      $a/*/order_line/item_id
  ```

  XML Column:

  ```
  SELECT item_id
  FROM order_line_item_side_tab line,
      order_side_tab order
  WHERE order.id = line.id and
      order_id = 3
  ```

  XML Collection:

  ```
  SELECT order_line_id
  FROM order_line_tab
  WHERE order_id = 3
  ```

  SQL Server:

  ```
  SELECT order_line_id
  FROM order_line_tab
  WHERE order_id = 3
  FOR XML AUTO
  ```

- **DC/MD_Q9** *Return the item ids of an order with id attribute value (4).*

  XQuery:

  ```
  for $a in input()/order[@id="4"]
  return
      $a//item_id
  ```

  XML Column:

  ```
  SELECT item_id
  FROM order_line_item_side_tab line,
      order_side_tab order
  WHERE order.id = line.id and
      order_id = 4
  ```

  XML Collection:

  ```
  SELECT order_line_id
  FROM order_line_tab
  WHERE order_id = 4
  ```

SQL Server:

```
SELECT order_line_id
FROM order_line_tab
WHERE order_id = 4
FOR XML AUTO
```

- **DC/MD_Q10** *List the orders (order id, order date and ship type), with total amount larger than a certain number (11000.0), ordered alphabetically by ship type.*

  XQuery:

```
for $a in input()/order
where $a/total gt 11000.0
order by $a/ship_type
return
    <Output>
        {$a/@id}
        {$a/order_date}
        {$a/ship_type}
    </Output>
```

  X-Hive:

```
for $a in input()/order
where $a/total gt 11000.0
return
    <Output>
        {$a/@id}
        {$a/order_date}
        {$a/ship_type}
    </Output>
sort by (ship_type)
```

  X-Hive/Rewrite:

```
for $a in input()/order
where $a/total gt double(11000.0)
return
<Output>
  {$a/@id}
  {$a/order_date}
  {$a/ship_type}
</Output>
sort by (ship_type)
```

  XML Column:

```
SELECT order_id,
    order_date,
    ship_type
FROM order_side_tab
WHERE total > 11000.0
ORDER BY ship_type
```

  XML Collection:

```
SELECT order_id,
    order_date,
    ship_type
```

```
FROM order_tab
WHERE total > 11000.0
ORDER BY ship_type
```

SQL Server:

```
SELECT order_id,
    order_date,
    ship_type
FROM order_tab
WHERE total > 11000.0
ORDER BY ship_type
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q11** *List the orders (order id, order date and order total), with total amount larger than a certain number (11000.0), in descending order by total amount.*

  XQuery:

```
for $a in input()/order
where $a/total gt 11000.0
order by $a/total descending
return
    <Output>
        {$a/@id}
        {$a/order_date}
        {$a/total}
    </Output>
```

  X-Hive:

```
for $a in input()/order
where $a/total gt 11000.0
return
    <Output>
        {$a/@id}
        {$a/order_date}
        {$a/total}
    </Output>
sort by ((total) descending)
```

  X-Hive/Rewrite:

```
for $a in input()/order
where $a/total gt double(11000.0)
return
<Output>
  {$a/@id}
  {$a/order_date}
  {$a/total}
</Output>
sort by ((total) descending)
```

  XML Column:

```
SELECT order_id,
    order_date,
    total
FROM order_side_tab
```

```
WHERE total > 11000.0
ORDER BY total DESC
```

XML Collection:

```
SELECT order_id,
    order_date,
    total
FROM order_tab
WHERE total > 11000.0
ORDER BY total DESC
```

SQL Server:

```
SELECT order_id,
    order_date,
    total
FROM order_tab
WHERE total > 11000.0
ORDER BY total DESC
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q12** *List all order lines of a certain order with id attribute value (5).*

XQuery:

```
for $a in input()/order[@id="5"]
return
    <Output>
        {$a/order_lines}
    </Output>
```

XML Column:

```
select order_line_id,
    item_id,
    quantity_of_item,
    discount_rate,
    special_instructions
from order_side_tab order,
    order_line_side_tab line,
    order_line_item_side_tab item,
    order_line_quan_side_tab quan,
    order_line_disc_side_tab disc,
    order_line_instr_side_tab instr
where order.id = line.id and
    order.id = item.id and
    order.id = quan.id and
    order.id = disc.id and
    order.id = instr.id and
    order.order_id = 5
```

XML Collection:

```
SELECT *
FROM order_line_tab
WHERE order_id = 5
```

SQL Server:

```
SELECT *
FROM order_line_tab
WHERE order_id = 5
FOR XML AUTO, ELEMENTS
```

- **DC/MD_Q14** *List the ids of orders that only have one order line.*

  XQuery:

```
for $a in input()/order
where empty($a/order_lines/order_line[2])
return
    <OneItemLine>
        {$a/@id}
    </OneItemLine>
```

  XML Column:

```
SELECT order_id
FROM order_side_tab order,
    (SELECT id,
    COUNT(order_line_id) number
    FROM order_line_side_tab
    GROUP BY id) temp
WHERE order.id = temp.id AND
    temp.number = 1
```

  XML Collection:

```
SELECT order_id
FROM
(SELECT order_id,
    COUNT(order_line_id) number
FROM order_line_tab
GROUP BY order_id) one_item_line
WHERE one_item_line.number = 1
```

  SQL Server:

```
SELECT order_id
FROM
(SELECT order_id,
    COUNT(order_line_id) number
FROM order_line_tab
GROUP BY order_id) one_item_line
WHERE one_item_line.number = 1
FOR XML AUTO
```

- **DC/MD_Q16** *Retrieve one whole order document with certain id attribute value (6).*

  XQuery:

```
for $a in input()/order[@id="6"]
return
    $a
```

  XML Column:

```
select db2xml.clob(order) order
from order_tab o,
    order_side_tab s
where o.id = s.id and
    s.order_id = 6
```

XML Collection:

```
SELECT *
FROM order_tab,
    order_line_tab
WHERE order_tab.order_id = order_line_tab.order_id AND
    order_tab.order_id = 6
```

SQL Server:

```
SELECT 1 tag,
  NULL AS parent,
  order_tab.order_id AS [order!1!id],
  customer_id AS [order!1!customer_id!element],
  order_date AS [order!1!order_date!element],
  subtotal AS [order!1!subtotal!element],
  tax AS [order!1!tax!element],
  total AS [order!1!total!element],
  ship_type AS [order!1!ship_type!element],
  ship_date AS [order!1!ship_date!element],
  bill_address_id AS [order!1!bill_address_id!element],
  ship_address_id AS [order!1!ship_address_id!element],
  order_status AS [order!1!order_status!element],
  NULL AS [credit_card_transaction!2!credit_card_type!element],
  NULL AS [credit_card_transaction!2!credit_card_number!element],
  NULL AS [credit_card_transaction!2!name_on_credit_card!element],
  NULL AS [credit_card_transaction!2!expiration_date!element],
  NULL AS [credit_card_transaction!2!authorization_id!element],
  NULL AS [credit_card_transaction!2!transaction_amount!element],
  NULL AS [credit_card_transaction!2!authorization_date!element],
  NULL AS [credit_card_transaction!2!transaction_country_id!element],
  NULL AS [order_line!3!id],
  NULL AS [order_line!3!item_id!element],
  NULL AS [order_line!3!quantity_of_item!element],
  NULL AS [order_line!3!discount_rate!element],
  NULL AS [order_line!3!special_instructions!element]
FROM order_tab
WHERE order_id = 6
UNION ALL
SELECT 2,
        1,
        order_id,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
```

```
            NULL,
            NULL,
            credit_card_type,
        credit_card_number,
        name_on_credit_card,
        expiration_date,
        authorization_id,
        transaction_amount,
        authorization_date,
        transaction_country_id,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL
    FROM order_tab
    WHERE order_id = 6
    UNION ALL
    SELECT 3,
            1,
            order_tab.order_id,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
            NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        order_line_id,
        item_id,
            quantity_of_item,
            discount_rate,
            special_instructions
    FROM order_tab, order_line_tab
    WHERE order_tab.order_id = order_line_tab.order_id AND
        order_tab.order_id = 6
    ORDER BY [order!1!id],tag
    FOR XML EXPLICIT
```

- **DC/MD_Q17** *Return the ids of authors whose biographies contain a certain word ("hockey").*

  XQuery:

```
for $a in input()/authors/author
```

```
where contains ($a/biography, "hockey")
return
<Output>
    {$a/@id}
</Output>
```

X-Hive/Rewrite:

```
for $a in input()/authors/author
where xhive:fts($a/biography, "hockey")
return
<Output>
  {$a/@id}
</Output>
```

XML Column:

```
SELECT DISTINCT(author_id)
FROM author_side_tab author,
author_bio_side_tab bio
WHERE author.id = bio.id and
author.dxx_seqno = bio.dxx_seqno and
biography like '%hockey%'
```

XML Collection:

```
SELECT author_id
FROM author_tab
WHERE biography like '%hockey%'
```

SQL Server:

```
SELECT author_id
FROM author_tab
WHERE biography like '%hockey%'
FOR XML AUTO
```

- **DC/MD_19** *For a particular order with id attribute value (7), get its customer name and phone, and its order status.*

XQuery:

```
for $order in input()/order,
    $cust in input()/customers/customer
where $order/customer_id = $cust/@id
    and $order/@id = "7"
return
    <Output>
        {$order/@id}
        {$order/order_status}
        {$cust/first_name}
        {$cust/last_name}
        {$cust/phone_number}
    </Output>
```

XML Column:

```
SELECT order_id,
    order_status,
    first_name,
```

```
    last_name,
    phone_number
FROM order_side_tab order,
customer_fname_side_tab first,
customer_lname_side_tab last,
customer_phone_side_tab phone,
customer_side_tab cust
WHERE cust.id = first.id AND
cust.id = last.id AND
cust.id = phone.id AND
cust.dxx_seqno = first.dxx_seqno AND
cust.dxx_seqno = last.dxx_seqno AND
cust.dxx_seqno = phone.dxx_seqno AND
order.customer_id = cust.customer_id AND
    order.order_id = 7
```

XML Collection:

```
SELECT order_id,
    order_status,
    first_name,
    last_name,
    phone_number
FROM order_tab, customer_tab
WHERE order_tab.customer_id = customer_tab.customer_id AND
    order_tab.order_id = 7
```

SQL Server:

```
SELECT order_id,
    order_status,
    first_name,
    last_name,
    phone_number
FROM order_tab, customer_tab
WHERE order_tab.customer_id = customer_tab.customer_id AND
    order_tab.order_id = 7
FOR XML AUTO, ELEMENTS
```

Appendix B

DB2 DADs

B.1 DAD for Dictionary (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>TCSD.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE dictionary SYSTEM "TCSD.dtd"</doctype>
        <root_node>
            <element_node name="dictionary">
                <RDB_node>
                    <table name="dictionary_tab" key="e_id"/>
                    <table name="hw_tab" key="hw"/>
                    <table name="pr_tab" key="pr"/>
                    <table name="pos_tab" key="pos"/>
                    <table name="vd_tab" key="vd"/>
                    <table name="vf_tab" key="vf"/>
                    <table name="et_cr_tab" key="cr"/>
                    <table name="s_tab" key="s_id"/>
                    <table name="def_cr_tab" key="cr"/>
                    <table name="q_tab" key="q_id"/>
                    <table name="qt_cr_tab" key="cr"/>
                    <table name="qt_i_tab" key=" i"/>
                    <table name="qt_b_tab" key="b"/>
                    <condition>dictionary_tab.e_id = hw_tab.e_id AND
                                dictionary_tab.e_id = pr_tab.e_id AND
                                dictionary_tab.e_id = pos_tab.e_id AND
                                dictionary_tab.e_id = vd_tab.e_id AND
                                dictionary_tab.e_id = vf_tab.e_id AND
                                dictionary_tab.e_id = et_cr_tab.e_id AND
                                dictionary_tab.e_id = s_tab.e_id AND
                                s_tab.e_id = def_cr_tab.e_id AND
                                s_tab.s_id = def_cr_tab.s_id AND
                                s_tab.e_id = q_tab.e_id AND
                                s_tab.s_id = q_tab.s_id
                    </condition>
                </RDB_node>
                <element_node name="e">
                    <attribute_node name="e_id">
                        <RDB_node>
                            <table name="dictionary_tab"/>
                            <column name="e_id" type="varchar(10)"/>
```

```
    </RDB_node>
</attribute_node>
<element_node name="hwg" multi_occurrence="YES">
    <element_node name="hw" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="hw_tab"/>
                <column name="hw" type="varchar(30)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="pr" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="pr_tab"/>
                <column name="pr" type="varchar(30)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="pos" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="pos_tab"/>
                <column name="pos" type="varchar(5)"/>
            </RDB_node>
        </text_node>
    </element_node>
</element_node>
<element_node name="vfl" multi_occurrence="YES">
    <element_node name=" vd" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="vd_tab"/>
                <column name="vd" type="integer"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="vf" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="vf_tab"/>
                <column name="vf" type="varchar(30)"/>
            </RDB_node>
        </text_node>
    </element_node>
</element_node>
<element_node name="et" multi_occurrence="YES">
    <element_node name="et_cr" multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="et_cr_tab"/>
                <column name="cr" type="varchar(10)"/>
            </RDB_node>
        </text_node>
    </element_node>
</element_node>
<element_node name="ss" multi_occurrence="YES">
    <element_node name="s" multi_occurrence="YES">
        <attribute_node name="s_id">
```

```
    <RDB_node>
        <table name="s_tab"/>
        <column name="s_id" type="integer"/>
    </RDB_node>
</attribute_node>
<element_node name="def" multi_occurrence="YES">
    <text_node>
        <RDB_node>
            <table name="s_tab"/>
            <column name="def"
            type="varchar(2500)"/>
        </RDB_node>
    </text_node>
    <element_node name="def_cr"
    multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="def_cr_tab"/>
                <column name="cr"
                type="varchar(10)"/>
            </RDB_node>
        </text_node>
    </element_node>
</element_node>
<element_node name="qp" multi_occurrence="YES">
    <element_node name="q" multi_occurrence="YES">
        <attribute_node name="q_id">
            <RDB_node>
                <table name="q_tab"/>
                <column name="q_id"
                type="integer"/>
            </RDB_node>
        </attribute_node>
        <element_node name="qd"
        multi_occurrence="YES">
            <text_node>
                <RDB_node>
                    <table name="q_tab"/>
                    <column name="qd"
                    type="integer"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="a"
        multi_occurrence="YES">
            <text_node>
                <RDB_node>
                    <table name="q_tab"/>
                    <column name="a"
                    type="varchar(60)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="w"
        multi_occurrence="YES">
            <text_node>
                <RDB_node>
                    <table name="q_tab"/>
                    <column name="w"
```

```
            type="varchar(60)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="bib"
multi_occurrence="YES">
    <text_node>
        <RDB_node>
            <table name="q_tab"/>
            <column name="bib"
            type="varchar(100)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="loc"
multi_occurrence="YES">
    <text_node>
        <RDB_node>
            <table name="q_tab"/>
            <column name="loc"
            type="varchar(60)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="qt"
multi_occurrence="YES">
    <text_node>
        <RDB_node>
            <table name="q_tab"/>
            <column name="qt"
            type="varchar(2500)"/>
        </RDB_node>
    </text_node>
    <element_node name="qt_cr"
    multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="qt_cr_tab"/>
                <column name="cr"
                type="varchar(100)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="i"
    multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="qt_i_tab"/>
                <column name="i"
                type="varchar(100)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="b"
    multi_occurrence="YES">
        <text_node>
            <RDB_node>
                <table name="qt_b_tab"/>
                <column name="b"
```

```
                                                    type="varchar(100)"/>
                                                </RDB_node>
                                            </text_node>
                                        </element_node>
                                    </element_node>
                                </element_node>
                            </element_node>
                        </element_node>
                    </element_node>
                </element_node>
            </root_node>
        </Xcollection>
</DAD>
```

B.2 DAD for Articles (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>TCMD.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn>
        <table name="article_side_tab">
            <column name="article_id" type="integer"
                path="/article/@id"
                multi_occurrence="NO"/>
            <column name="title" type="varchar(300)"
                path="/article/prolog/title"
                multi_occurrence="NO"/>
            <column name="date" type="date"
                path="/article/prolog/dateline/date"
                multi_occurrence="NO"/>
            <column name="country" type="varchar(50)"
                path="/article/prolog/dateline/country"
                multi_occurrence="NO"/>
            <column name="genre" type="varchar(50)"
                path="/article/prolog/genre"
                multi_occurrence="NO"/>
        </table>
        <table name="author_side_tab">
            <column name="name" type="varchar(50)"
                path="/article/prolog/authors/author/name"
                multi_occurrence="YES"/>
        </table>
        <table name="section_side_tab">
            <column name="heading" type="varchar(200)"
                path="/article/body/section/@heading"
                multi_occurrence="YES"/>
        </table>
        <table name="p_side_tab">
            <column name="p" type="varchar(20000)"
                path="/article//p"
                multi_occurrence="YES"/>
        </table>
        <table name="phone_side_tab">
            <column name="phone" type="varchar(30)"
                path="/article//phone"
                multi_occurrence="YES"/>
        </table>
```

```
        <table name="email_side_tab">
            <column name="email" type="varchar(100)"
                path="/article//email"
                multi_occurrence="YES"/>
        </table>
        <table name="ref_side_tab">
            <column name="a_id" type="integer"
                path="/article/epilog/references/a_id"
                multi_occurrence="YES"></column>
        </table>
    </Xcolumn>
</DAD>
```

B.3 DAD for Article (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>TCMD.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE article SYSTEM "D:\xbench\schemas\TCMD.dtd"</doctype>
        <root_node>
            <element_node name="article">
                <RDB_node>
                    <table name="article_tab" key="article_id"/>
                    <table name="article_author_tab"/>
                    <table name="keyword_tab"/>
                    <table name="body_p_tab"/>
                    <table name="ack_tab"/>
                    <table name="ref_tab"/>
                    <table name="body_sec_tab"/>
                    <table name="sec_p_tab"/>
                    <table name="sec_subsec_tab"/>
                    <table name="subsec_p_tab"/>
                    <table name="subsec_subsubsec_tab"/>
                    <table name="subsubsec_p_tab"/>
                    <table name="subsubsec_subsubsubsec_tab"/>
                    <table name="subsubsubsec_p_tab"/>
                    <condition>article_tab.article_id=
                        article_author_tab.article_id AND
                        article_tab.article_id=keyword_tab.article_id AND
                        article_tab.article_id=body_p_tab.article_id AND
                        article_tab.article_id=ack_tab.article_id AND
                        article_tab.article_id=ref_tab.article_id AND
                        article_tab.article_id=body_sec_tab.article_id AND
                        article_tab.article_id=sec_p_tab.article_id AND
                        article_tab.article_id=subsec_p_tab.article_id AND
                        article_tab.article_id=subsubsec_p_tab.article_id AND
                        article_tab.article_id=subsubsubsec_p_tab.article_id AND
                        article_tab.article_id=sec_subsec_tab.article_id AND
                        article_tab.article_id=subsec_subsubsec_tab.article_id AND
                        article_tab.article_id=
                        subsubsec_subsubsubsec_tab.article_id
                        </condition>
                </RDB_node>
                <attribute_node name="lang">
                    <RDB_node>
                        <table name="article_tab"/>
```

```
        <column name="lang" type="char(2)"/>
    </RDB_node>
</attribute_node>
<attribute_node name="id">
    <RDB_node>
        <table name="article_tab"/>
        <column name="article_id" type="integer"/>
    </RDB_node>
</attribute_node>
<element_node name="prolog">
    <element_node name="title">
        <text_node>
            <RDB_node>
                <table name="article_tab"/>
                <column name="title" type="varchar(252)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="authors">
        <element_node name="author" multi_occurrence="YES">
            <element_node name="name">
                <text_node>
                    <RDB_node>
                        <table name="article_author_tab"/>
                        <column name="name" type="varchar(60)"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="contact">
                <element_node name="email">
                    <text_node>
                        <RDB_node>
                            <table name="article_author_tab"/>
                            <column name="email"
                            type="varchar(50)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="phone">
                    <text_node>
                        <RDB_node>
                            <table name="article_author_tab"/>
                            <column name="phone"
                            type="varchar(25)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
            </element_node>
        </element_node>
    </element_node>
    <element_node name="dateline">
        <element_node name="city">
            <text_node>
                <RDB_node>
                    <table name="article_tab"/>
                    <column name="city" type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
```

```
            <element_node name="country">
                <text_node>
                    <RDB_node>
                        <table name="article_tab"/>
                        <column name="country" type="varchar(25)"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="date">
                <text_node>
                    <RDB_node>
                        <table name="article_tab"/>
                        <column name="date" type="date"/>
                    </RDB_node>
                </text_node>
            </element_node>
        </element_node>
        <element_node name="genre">
            <text_node>
                <RDB_node>
                    <table name="article_tab"/>
                    <column name="genre" type="varchar(20)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="keywords">
            <element_node name="keyword">
                <text_node>
                    <RDB_node>
                        <table name="keyword_tab"/>
                        <column name="keyword" type="varchar(96)"/>
                    </RDB_node>
                </text_node>
            </element_node>
        </element_node>
    </element_node>
    <element_node name="body">
        <element_node name="abstract">
            <element_node name="p">
                <text_node>
                    <RDB_node>
                        <table name="body_p_tab"/>
                        <column name="p" type="varchar(20000)"/>
                    </RDB_node>
                </text_node>
            </element_node>
        </element_node>
        <element_node name="section">
            <attribute_node name="heading">
                <RDB_node>
                    <table name="body_sec_tab"/>
                    <column name="sec_heading" type="varchar(200)"/>
                </RDB_node>
            </attribute_node>
            <element_node name="p">
                <text_node>
                    <RDB_node>
                        <table name="sec_p_tab"/>
                        <column name="p" type="varchar(20000)"/>
```

```
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="subsec">
            <attribute_node name="heading">
                <RDB_node>
                    <table name="sec_subsec_tab"/>
                    <column name="subsec_heading"
                    type="varchar(200)"/>
                </RDB_node>
            </attribute_node>
            <element_node name="p">
                <text_node>
                    <RDB_node>
                        <table name="subsec_p_tab"/>
                        <column name="p" type="varchar(20000)"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="subsec">
                <attribute_node name="heading">
                    <RDB_node>
                        <table name="subsec_subsubsec_tab"/>
                        <column name="subsubsec_heading"
                        type="varchar(200)"/>
                    </RDB_node>
                </attribute_node>
                <element_node name="p">
                    <text_node>
                        <RDB_node>
                            <table name="subsubsec_p_tab"/>
                            <column name="p" type="varchar(20000)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="subsec">
                    <attribute_node name="heading">
                        <RDB_node>
                            <table name=
                            "subsubsec_subsubsubsec_tab"/>
                            <column name="subsubsubsec_heading"
                            type="varchar(200)"/>
                        </RDB_node>
                    </attribute_node>
                    <element_node name="p">
                        <text_node>
                            <RDB_node>
                                <table name="subsubsubsec_p_tab"/>
                                <column name="p"
                                type="varchar(20000)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                </element_node>
            </element_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="epilog">
```

```
                    <element_node name="acknowledgements">
                        <element_node name="pa">
                            <text_node>
                                <RDB_node>
                                    <table name="ack_tab"/>
                                    <column name="pa" type="varchar(20000)"/>
                                </RDB_node>
                            </text_node>
                        </element_node>
                    </element_node>
                    <element_node name="references">
                        <element_node name="a_id">
                            <text_node>
                                <RDB_node>
                                    <table name="ref_tab"/>
                                    <column name="a_id" type="integer"/>
                                </RDB_node>
                            </text_node>
                        </element_node>
                    </element_node>
                </element_node>
            </element_node>
        </root_node>
    </Xcollection>
</DAD>
```

B.4 DAD for Catalog (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCSD.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE catalog SYSTEM "DCSD.dtd"</doctype>
        <root_node>
            <element_node name="catalog">
                <RDB_node>
                    <table name="catalog_tab" key="item_id"/>
                    <table name="catalog_related_item_tab"
                        key="item_id related_item_id"/>
                    <table name="catalog_author_tab"
                        key="first_name middle_name last_name"/>
                    <table name="catalog_author_address_tab"
                        key="street_address"/>
                    <table name="catalog_publisher_address_tab"
                        key="street_address"/>
                    <condition>catalog_tab.item_id=catalog_author_tab.item_id AND
                    catalog_tab.item_id=catalog_publisher_address_tab.item_id AND
                    catalog_author_tab.item_id=
                        catalog_author_address_tab.item_id AND
                    catalog_author_tab.first_name=
                        catalog_author_address_tab.first_name AND
                    catalog_author_tab.middle_name=
                        catalog_author_address_tab.middle_name AND
                    catalog_author_tab.last_name=
                        catalog_author_address_tab.last_name</condition>
                </RDB_node>
                <element_node name="item">
```

```
<attribute_node name="id">
    <RDB_node>
        <table name="catalog_tab"/>
        <column name="item_id" type="varchar(10)"/>
    </RDB_node>
</attribute_node>
<element_node name="title">
    <text_node>
        <RDB_node>
            <table name="catalog_tab"/>
            <column name="title" type="varchar(100)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="authors">
    <element_node name="author" multi_occurrence="YES">
        <element_node name="first_name">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="first_name"
                    type="varchar(30)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="middle_name">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="middle_name"
                    type="varchar(30)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="last_name">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="last_name"
                    type="varchar(30)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="date_of_birth">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="date_of_birth" type="date"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="biography">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="biography"
                    type="varchar(500)"/>
                </RDB_node>
            </text_node>
```

```
</element_node>
<element_node name="contact_information">
    <element_node name="mailing_address">
        <element_node name="street_information">
            <element_node name="street_address">
                <text_node>
                <RDB_node>
                    <table name=
                    "catalog_author_address_tab"/>
                    <column name="street_address"
                        type="varchar(40)"/>
                </RDB_node>
                </text_node>
            </element_node>
        </element_node>
        <element_node name="name_of_city">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="name_of_city"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="name_of_state">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="name_of_state"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="zip_code">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="zip_code"
                        type="varchar(7)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="name_of_country">
            <text_node>
                <RDB_node>
                    <table name="catalog_author_tab"/>
                    <column name="name_of_country"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
    <element_node name="phone_number">
        <text_node>
            <RDB_node>
                <table name="catalog_author_tab"/>
                <column name="phone_number"
                    type="varchar(25)"/>
            </RDB_node>
```

```
                </text_node>
            </element_node>
            <element_node name="email_address">
                <text_node>
                    <RDB_node>
                        <table name="catalog_author_tab"/>
                        <column name="email_address"
                            type="varchar(50)"/>
                    </RDB_node>
                </text_node>
            </element_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="date_of_release">
    <text_node>
        <RDB_node>
            <table name="catalog_tab"/>
            <column name="date_of_release" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="publisher">
    <element_node name="p_name">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="publisher_name"
                    type="varchar(100)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="contact_information">
        <element_node name="mailing_address">
            <element_node name="street_information">
                <element_node name="p_street_address">
                    <text_node>
                    <RDB_node>
                        <table name=
                        "catalog_publisher_address_tab"/>
                        <column name="street_address"
                            type="varchar(40)"/>
                    </RDB_node>
                    </text_node>
                </element_node>
            </element_node>
            <element_node name="name_of_city">
                <text_node>
                    <RDB_node>
                        <table name="catalog_tab"/>
                        <column name="publisher_name_of_city"
                            type="varchar(25)"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="name_of_state">
                <text_node>
                    <RDB_node>
                        <table name="catalog_tab"/>
```

```
                <column name="publisher_name_of_state"
                    type="varchar(25)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="zip_code">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="publisher_zip_code"
                    type="varchar(7)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="country">
        <element_node name="country_name">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name=
                        "publisher_country_name"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="exchange_rate">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name=
                        "publisher_country_ex_rate"
                        type="decimal(10,4)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="currency">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name=
                        "publisher_country_currency"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="FAX_number">
    <text_node>
        <RDB_node>
            <table name="catalog_tab"/>
            <column name="publisher_FAX_number"
                type="varchar(25)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="phone_number">
    <text_node>
        <RDB_node>
```

```
                    <table name="catalog_tab"/>
                    <column name="publisher_phone_number"
                        type="varchar(25)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="web_site">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="publisher_web_site"
                        type="varchar(50)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="subject">
    <text_node>
        <RDB_node>
            <table name="catalog_tab"/>
            <column name="subject" type="varchar(20)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="description">
    <text_node>
        <RDB_node>
            <table name="catalog_tab"/>
            <column name="description" type="varchar(500)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="related_items">
    <element_node name="related_item">
        <attribute_node name="c_id">
            <RDB_node>
                <table name="catalog_related_item_tab"/>
                <column name="item_id" type="varchar(10)"/>
            </RDB_node>
        </attribute_node>
        <element_node name="item_id">
            <text_node>
                <RDB_node>
                    <table name="catalog_related_item_tab"/>
                    <column name="related_item_id"
                        type="varchar(10)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="media">
    <element_node name="thumbnail">
        <element_node name="data">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="thumbnail_data"
```

```
                        type="varchar(1)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
    <element_node name="image">
        <element_node name="data">
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="image_data"
                        type="varchar(1)"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
</element_node>
<element_node name="pricing">
    <element_node name="suggested_retail_price">
        <attribute_node name="suggested_retail_price_currency">
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="suggested_retail_price_cur"
                    type="varchar(25)"/>
            </RDB_node>
        </attribute_node>
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="suggested_retail_price"
                    type="decimal(8,2)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="cost">
        <attribute_node name="cost_currency">
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="cost_currency"
                    type="varchar(25)"/>
            </RDB_node>
        </attribute_node>
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="cost" type="decimal(8,2)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="when_is_available">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="when_is_available"
                type="date"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="quantity_in_stock">
```

```
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="quantity_in_stock"
                type="integer"/>
            </RDB_node>
        </text_node>
    </element_node>
</element_node>
<element_node name="attributes">
    <element_node name="ISBN">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="ISBN" type="varchar(30)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="number_of_pages">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="number_of_pages"
                type="integer"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="type_of_book">
        <text_node>
            <RDB_node>
                <table name="catalog_tab"/>
                <column name="type_of_book"
                type="varchar(20)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="size_of_book">
        <element_node name="length">
            <attribute_node name="length_unit">
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="length_unit"
                    type="varchar(10)"/>
                </RDB_node>
            </attribute_node>
            <text_node>
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="length"
                    type="decimal(5,1)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="width">
            <attribute_node name="width_unit">
                <RDB_node>
                    <table name="catalog_tab"/>
                    <column name="width_unit"
                    type="varchar(10)"/>
```

```
                                    </RDB_node>
                                </attribute_node>
                                <text_node>
                                    <RDB_node>
                                        <table name="catalog_tab"/>
                                        <column name="width"
                                        type="decimal(5,1)"/>
                                    </RDB_node>
                                </text_node>
                            </element_node>
                            <element_node name="height">
                                <attribute_node name="height_unit">
                                    <RDB_node>
                                        <table name="catalog_tab"/>
                                        <column name="height_unit"
                                            type="varchar(10)"/>
                                    </RDB_node>
                                </attribute_node>
                                <text_node>
                                    <RDB_node>
                                        <table name="catalog_tab"/>
                                        <column name="height"
                                        type="decimal(5,1)"/>
                                    </RDB_node>
                                </text_node>
                            </element_node>
                        </element_node>
                    </element_node>
                </element_node>
            </element_node>
        </root_node>
    </Xcollection>
</DAD>
```

B.5 DAD for Item (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDItem.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn>
        <table name="item_side_tab">
            <column name="item_id"  type="integer"
                path="/items/item/@id" multi_occurrence="YES">
            </column>
        </table>
    </Xcolumn>
</DAD>
```

B.6 DAD for Item (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDItem.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE item SYSTEM "D:\xbench\schemas\DCMDItem.dtd"</doctype>
        <root_node>
```

```
<element_node name="items">
    <RDB_node>
        <table name="item_tab" key="item_id"/>
        <table name="related_item_tab"/>
        <condition>
        item_tab.item_id=related_item_tab.item_id
        </condition>
    </RDB_node>
    <element_node name="item">
        <attribute_node name="id">
            <RDB_node>
                <table name="item_tab"/>
                <column name="item_id" type="integer"/>
            </RDB_node>
        </attribute_node>
        <element_node name="title">
            <text_node>
                <RDB_node>
                    <table name="item_tab"/>
                    <column name="title" type="varchar(80)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="author_id">
            <text_node>
                <RDB_node>
                    <table name="item_tab"/>
                    <column name="author_id" type="integer"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="date_of_release">
            <text_node>
                <RDB_node>
                    <table name="item_tab"/>
                    <column name="date_of_release" type="date"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="name_of_publisher">
            <text_node>
                <RDB_node>
                    <table name="item_tab"/>
                    <column name="name_of_publisher"
                        type="varchar(60)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="subject">
            <text_node>
                <RDB_node>
                    <table name="item_tab"/>
                    <column name="subject" type="varchar(15)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="description">
            <text_node>
                <RDB_node>
```

```
            <table name="item_tab"/>
            <column name="description" type="varchar(500)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="related_item_id">
    <text_node>
        <RDB_node>
            <table name="related_item_tab"/>
            <column name="related_item_id" type="integer"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="thumbnail">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="thumbnail" type="varchar(1)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="image">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="image" type="varchar(1)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="suggested_retail_price">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="suggested_retail_price"
            type="decimal(6,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="cost">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="cost" type="decimal(6,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="when_is_available">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="when_is_available" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="quantity_in_stock">
    <text_node>
        <RDB_node>
            <table name="item_tab"/>
            <column name="quantity_in_stock" type="integer"/>
```

```
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="ISBN">
                    <text_node>
                        <RDB_node>
                            <table name="item_tab"/>
                            <column name="ISBN" type="char(14)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="number_of_pages">
                    <text_node>
                        <RDB_node>
                            <table name="item_tab"/>
                            <column name="number_of_pages" type="integer"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="type_of_book">
                    <text_node>
                        <RDB_node>
                            <table name="item_tab"/>
                            <column name="type_of_book" type="varchar(15)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="size_of_book">
                    <text_node>
                        <RDB_node>
                            <table name="item_tab"/>
                            <column name="size_of_book" type="varchar(17)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
            </element_node>
        </element_node>
    </root_node>
</Xcollection>
</DAD>
```

B.7 DAD for Order (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDOrd.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn>
        <table name="order_side_tab">
            <column name="order_id" type="integer"
                path="/order/@id" multi_occurrence="NO"/>
            <column name="customer_id" type="integer"
                path="/order/customer_id" multi_occurrence="NO"/>
            <column name="total" type="decimal(10,2)"
                path="/order/total" multi_occurrence="NO"/>
            <column name="order_date" type="date"
                path="/order/order_date" multi_occurrence="NO"/>
            <column name="ship_type" type="varchar(10)"
                path="/order/ship_type" multi_occurrence="NO"/>
```

```
            <column name="order_status" type="varchar(10)"
                path="/order/order_status" multi_occurrence="NO"/>
        </table>
        <table name="order_line_side_tab">
            <column name="order_line_id" type="integer"
                path="/order//order_line/@id" multi_occurrence="YES"/>
        </table>
        <table name="order_line_disc_side_tab">
            <column name="discount_rate" type="decimal(5,2)"
                path="/order//discount_rate" multi_occurrence="YES"/>
        </table>
        <table name="order_line_item_side_tab">
            <column name="item_id" type="integer"
                path="/order//item_id" multi_occurrence="YES"/>
        </table>
        <table name="order_line_quan_side_tab">
            <column name="quantity_of_item" type="integer"
                path="/order//quantity_of_item" multi_occurrence="YES"/>
        </table>
        <table name="order_line_instr_side_tab">
            <column name="special_instructions" type="varchar(100)"
                path="/order//special_instructions" multi_occurrence="YES"/>
        </table>

    </Xcolumn>
</DAD>
```

B.8 DAD for Order (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDOrd.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE order SYSTEM "D:\xbench\schemas\DCMDOrd.dtd"</doctype>
        <root_node>
            <element_node name="order">
                <RDB_node>
                    <table name="order_tab" key="order_id"/>
                    <table name="order_line_tab"/>
                    <condition>
                    order_tab.order_id=order_line_tab.order_id
                    </condition>
                </RDB_node>
                <attribute_node name="id">
                    <RDB_node>
                        <table name="order_tab"/>
                        <column name="order_id" type="integer"/>
                    </RDB_node>
                </attribute_node>
                <element_node name="customer_id">
                    <text_node>
                        <RDB_node>
                            <table name="order_tab"/>
                            <column name="customer_id" type="integer"/>
                        </RDB_node>
                    </text_node>
                </element_node>
```

```
<element_node name="order_date">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="order_date" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="subtotal">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="subtotal" type="decimal(6,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="tax">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="taxl" type="decimal(5,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="total">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="total" type="decimal(7,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="ship_type">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="ship_type" type="varchar(10)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="ship_date">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="ship_date" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="bill_address_id">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="bill_address_id" type="integer"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="ship_address_id">
    <text_node>
        <RDB_node>
```

```
            <table name="order_tab"/>
            <column name="ship_address_id" type="integer"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="order_status">
    <text_node>
        <RDB_node>
            <table name="order_tab"/>
            <column name="order_status" type="varchar(10)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="credit_card_transaction">
    <element_node name="credit_card_type">
        <text_node>
            <RDB_node>
                <table name="order_tab"/>
                <column name="credit_card_type"
                type="varchar(10)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="credit_card_number">
        <text_node>
            <RDB_node>
                <table name="order_tab"/>
                <column name="credit_card_number"
                type="varchar(16)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="name_on_credit_card">
        <text_node>
            <RDB_node>
                <table name="order_tab"/>
                <column name="name_on_credit_card"
                type="varchar(30)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="expiration_date">
        <text_node>
            <RDB_node>
                <table name="order_tab"/>
                <column name="expiration_date" type="date"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="authorization_id">
        <text_node>
            <RDB_node>
                <table name="order_tab"/>
                <column name="authorization_id"
                    type="varchar(15)"/>
            </RDB_node>
        </text_node>
    </element_node>
    <element_node name="transaction_amount">
```

```
            <text_node>
                <RDB_node>
                    <table name="order_tab"/>
                    <column name="transaction_amount"
                        type="decimal(7,2)"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="authorization_date">
            <text_node>
                <RDB_node>
                    <table name="order_tab"/>
                    <column name="authorization_date" type="date"/>
                </RDB_node>
            </text_node>
        </element_node>
        <element_node name="transaction_country_id">
            <text_node>
                <RDB_node>
                    <table name="order_tab"/>
                    <column name="transaction_country_id"
                        type="integer"/>
                </RDB_node>
            </text_node>
        </element_node>
    </element_node>
    <element_node name="order_lines">
        <element_node name="order_line">
            <attribute_node name="id">
                <RDB_node>
                    <table name="order_line_tab"/>
                    <column name="order_line_id" type="integer"/>
                </RDB_node>
            </attribute_node>
            <element_node name="item_id">
                <text_node>
                    <RDB_node>
                        <table name="order_line_tab"/>
                        <column name="item_id" type="integer"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="quantity_of_item">
                <text_node>
                    <RDB_node>
                        <table name="order_line_tab"/>
                        <column name="quantity_of_item"
                            type="integer"/>
                    </RDB_node>
                </text_node>
            </element_node>
            <element_node name="discount_rate">
                <text_node>
                    <RDB_node>
                        <table name="order_line_tab"/>
                        <column name="discount_rate"
                            type="decimal(3,2)"/>
                    </RDB_node>
                </text_node>
```

```
                        </element_node>
                        <element_node name="special_instructions">
                            <text_node>
                                <RDB_node>
                                    <table name="order_line_tab"/>
                                    <column name="special_instructions"
                                        type="varchar(100)"/>
                                </RDB_node>
                            </text_node>
                        </element_node>
                    </element_node>
                </element_node>
            </element_node>
        </root_node>
    </Xcollection>
</DAD>
```

B.9 DAD for Customer (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDCust.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn>
        <table name="customer_side_tab">
            <column name="customer_id" type="integer"
                path="/customers/customer/@id" multi_occurrence="YES"/>
        </table>
        <table name="customer_fname_side_tab">
            <column name="first_name" type="varchar(50)"
                path="/customers/customer/first_name" multi_occurrence="YES"/>
        </table>
        <table name="customer_lname_side_tab">
            <column name="last_name" type="varchar(50)"
                path="/customers/customer/last_name" multi_occurrence="YES"/>
        </table>
        <table name="customer_phone_side_tab">
            <column name="phone_number" type="varchar(30)"
                path="/customers/customer/phone_number" multi_occurrence="YES"/>
        </table>
    </Xcolumn>
</DAD>
```

B.10 DAD for Customer (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDCust.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE customer SYSTEM
            "D:\xbench\schemas\DCMDCust.dtd"</doctype>
        <root_node>
            <element_node name="customers">
                <RDB_node>
                    <table name="customer_tab" key="customer_id"/>
                </RDB_node>
                <element_node name="customer">
```

```
<attribute_node name="id">
    <RDB_node>
        <table name="customer_tab"/>
        <column name="customer_id" type="integer"/>
    </RDB_node>
</attribute_node>
<element_node name="user_name">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="user_name" type="varchar(20)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="password">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="password" type="varchar(20)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="first_name">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="first_name" type="varchar(15)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="last_name">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="last_name" type="varchar(15)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="address_id">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="address_id" type="integer"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="phone_number">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="phone_number" type="varchar(16)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="email_address">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="email_address" type="varchar(50)"/>
```

```
        </RDB_node>
    </text_node>
</element_node>
<element_node name="date_of_registration">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="date_of_registration" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="date_of_last_visit">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="date_of_last_visit" type="date"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="start_of_current_session">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="start_of_current_session"
                type="char(29)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="current_session_expiry">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="current_session_expiry"
                type="char(29)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="discount_rate">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="discount_rate" type="decimal(3,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="balance">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="balance" type="decimal(3,2)"/>
        </RDB_node>
    </text_node>
</element_node>
<element_node name="YTD_payment">
    <text_node>
        <RDB_node>
            <table name="customer_tab"/>
            <column name="YTD_payment" type="decimal(5,2)"/>
        </RDB_node>
```

```
                    </text_node>
                </element_node>
                <element_node name="birth_date">
                    <text_node>
                        <RDB_node>
                            <table name="customer_tab"/>
                            <column name="birth_date" type="date"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="miscellaneous_information">
                    <text_node>
                        <RDB_node>
                            <table name="customer_tab"/>
                            <column name="miscellaneous_information"
                                type="varchar(500)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
            </element_node>
        </element_node>
    </root_node>
</Xcollection>
</DAD>
```

B.11 DAD for Author (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDAuth.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn>
        <table name="author_side_tab">
            <column name="author_id" type="integer"
                path="/authors/author/@id" multi_occurrence="YES"/>
        </table>
        <table name="author_bio_side_tab">
            <column name="biography" type="varchar(500)"
                path="/authors/author/biography" multi_occurrence="YES"/>
        </table>
    </Xcolumn>
</DAD>
```

B.12 DAD for Author (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDAuth.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE author SYSTEM "D:\xbench\schemas\DCMDAuth.dtd"</doctype>
        <root_node>
            <element_node name="authors">
                <RDB_node>
                    <table name="author_tab" key="author_id"/>
                </RDB_node>
                <element_node name="author">
                    <attribute_node name="id">
```

```
                    <RDB_node>
                        <table name="author_tab"/>
                        <column name="author_id" type="integer"/>
                    </RDB_node>
                </attribute_node>
                <element_node name="first_name">
                    <text_node>
                        <RDB_node>
                            <table name="author_tab"/>
                            <column name="first_name" type="varchar(20)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="middle_name">
                    <text_node>
                        <RDB_node>
                            <table name="author_tab"/>
                            <column name="middle_name" type="varchar(20)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="last_name">
                    <text_node>
                        <RDB_node>
                            <table name="author_tab"/>
                            <column name="last_name" type="varchar(30)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="date_of_birth">
                    <text_node>
                        <RDB_node>
                            <table name="author_tab"/>
                            <column name="date_of_birth" type="date"/>
                        </RDB_node>
                    </text_node>
                </element_node>
                <element_node name="biography">
                    <text_node>
                        <RDB_node>
                            <table name="author_tab"/>
                            <column name="biography" type="varchar(500)"/>
                        </RDB_node>
                    </text_node>
                </element_node>
            </element_node>
        </element_node>
    </root_node>
</Xcollection>
</DAD>
```

B.13 DAD for Country (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDCoun.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn/>
</DAD>
```

B.14 DAD for Country (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDCoun.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE country SYSTEM "D:\xbench\schemas\DCMDCoun.dtd"</doctype>
        <root_node>
            <element_node name="countries">
                <RDB_node>
                    <table name="country_tab" key="country_id"/>
                </RDB_node>
                <element_node name="country">
                    <attribute_node name="id">
                        <RDB_node>
                            <table name="country_tab"/>
                            <column name="country_id" type="integer"/>
                        </RDB_node>
                    </attribute_node>
                    <element_node name="name">
                        <text_node>
                            <RDB_node>
                                <table name="country_tab"/>
                                <column name="name" type="varchar(20)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="exchange_rate">
                        <text_node>
                            <RDB_node>
                                <table name="country_tab"/>
                                <column name="exchange_rate" type="decimal(12,6)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="currency">
                        <text_node>
                            <RDB_node>
                                <table name="country_tab"/>
                                <column name="currency" type="varchar(20)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                </element_node>
            </element_node>
        </root_node>
    </Xcollection>
</DAD>
```

B.15 DAD for Address (XML Column)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDAddr.dtd</dtdid>
    <validation>NO</validation>
    <Xcolumn/>
</DAD>
```

B.16 DAD for Address (XML Collection)

```
<?xml version="1.0"?> <!DOCTYPE DAD SYSTEM "dad.dtd"> <DAD>
    <dtdid>DCMDAddr.dtd</dtdid>
    <validation>NO</validation>
    <Xcollection>
        <prolog>?xml version="1.0"?</prolog>
        <doctype>!DOCTYPE address SYSTEM "D:\xbench\schemas\DCMDAddr.dtd"</doctype>
        <root_node>
            <element_node name="addresses">
                <RDB_node>
                    <table name="address_tab" key="address_id"/>
                    <table name="street_address_tab"/>
                    <condition>address_tab.address_id=street_address_tab.address_id
                    </condition>
                </RDB_node>
                <element_node name="address">
                    <attribute_node name="id">
                        <RDB_node>
                            <table name="address_tab"/>
                            <column name="address_id" type="integer"/>
                        </RDB_node>
                    </attribute_node>
                    <element_node name="street_address">
                        <text_node>
                            <RDB_node>
                                <table name="street_address_tab"/>
                                <column name="street_address" type="varchar(40)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="name_of_city">
                        <text_node>
                            <RDB_node>
                                <table name="address_tab"/>
                                <column name="name_of_city" type="varchar(30)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="name_of_state">
                        <text_node>
                            <RDB_node>
                                <table name="address_tab"/>
                                <column name="name_of_state" type="varchar(20)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="zip_code">
                        <text_node>
                            <RDB_node>
                                <table name="address_tab"/>
                                <column name="zip_code" type="varchar(10)"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                    <element_node name="country_id">
                        <text_node>
                            <RDB_node>
                                <table name="address_tab"/>
```

```
                                <column name="country_id" type="integer"/>
                            </RDB_node>
                        </text_node>
                    </element_node>
                </element_node>
            </element_node>
        </root_node>
    </Xcollection>
</DAD>
```

Appendix C

DB2 DDLs

C.1 DB2 DDL for TCSD Class

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."DICTIONARY_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."DICTIONARY_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."DICTIONARY_TAB"

ALTER TABLE "BBYAO   "."DICTIONARY_TAB"
    ADD PRIMARY KEY
        ("E_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."HW_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."HW_TAB"  (
          "HW" VARCHAR(30) NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."HW_TAB"

ALTER TABLE "BBYAO   "."HW_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "HW");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."PR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."PR_TAB"  (
          "PR" VARCHAR(30) NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
```

```
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."PR_TAB"

ALTER TABLE "BBYAO   "."PR_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "PR");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."POS_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."POS_TAB"  (
          "POS" VARCHAR(5) NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."POS_TAB"

ALTER TABLE "BBYAO   "."POS_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "POS");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."VD_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."VD_TAB"  (
          "VD" INTEGER NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."VD_TAB"

ALTER TABLE "BBYAO   "."VD_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "VD");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."VF_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."VF_TAB"  (
          "VF" VARCHAR(30) NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."VF_TAB"

ALTER TABLE "BBYAO   "."VF_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "VF");
```

```
-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ET_CR_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ET_CR_TAB"  (
          "CR" VARCHAR(10) NOT NULL ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."ET_CR_TAB"

ALTER TABLE "BBYAO   "."ET_CR_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "CR");


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."S_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."S_TAB"  (
          "S_ID" INTEGER NOT NULL ,
          "DEF" VARCHAR(2500) ,
          "E_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."S_TAB"

ALTER TABLE "BBYAO   "."S_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID");


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."DEF_CR_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."DEF_CR_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL ,
          "S_ID" INTEGER NOT NULL ,
          "CR" VARCHAR(10) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."DEF_CR_TAB"

ALTER TABLE "BBYAO   "."DEF_CR_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID",
        "CR");

-------------------------------------------------
-- DDL Statements for table "BBYAO   "."Q_TAB"
-------------------------------------------------
```

```
 CREATE TABLE "BBYAO   "."Q_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL ,
          "S_ID" INTEGER NOT NULL ,
          "Q_ID" INTEGER NOT NULL ,
          "QD" INTEGER ,
          "A" VARCHAR(60) ,
          "W" VARCHAR(60) ,
          "BIB" VARCHAR(100) ,
          "LOC" VARCHAR(60) ,
          "QT" VARCHAR(2500) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."Q_TAB"

ALTER TABLE "BBYAO   "."Q_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID",
        "Q_ID");

------------------------------------------------
-- DDL Statements for table "BBYAO   "."QT_CR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."QT_CR_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL ,
          "S_ID" INTEGER NOT NULL ,
          "Q_ID" INTEGER NOT NULL ,
          "CR" VARCHAR(100) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."QT_CR_TAB"

ALTER TABLE "BBYAO   "."QT_CR_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID",
        "Q_ID",
        "CR");

------------------------------------------------
-- DDL Statements for table "BBYAO   "."QT_I_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."QT_I_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL ,
          "S_ID" INTEGER NOT NULL ,
          "Q_ID" INTEGER NOT NULL ,
          "I" VARCHAR(100) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."QT_I_TAB"

ALTER TABLE "BBYAO   "."QT_I_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID",
        "Q_ID",
        "I");
```

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."QT_B_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."QT_B_TAB"  (
          "E_ID" VARCHAR(10) NOT NULL ,
          "S_ID" INTEGER NOT NULL ,
          "Q_ID" INTEGER NOT NULL ,
          "B" VARCHAR(100) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."QT_B_TAB"

ALTER TABLE "BBYAO   "."QT_B_TAB"
    ADD PRIMARY KEY
        ("E_ID",
        "S_ID",
        "Q_ID",
        "B");


-- DDL Statements for foreign keys on Table "BBYAO   "."HW_TAB"

ALTER TABLE "BBYAO   "."HW_TAB"
    ADD CONSTRAINT "SQL030417184459711" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."PR_TAB"

ALTER TABLE "BBYAO   "."PR_TAB"
    ADD CONSTRAINT "SQL030417184459861" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."POS_TAB"

ALTER TABLE "BBYAO   "."POS_TAB"
    ADD CONSTRAINT "SQL030417184500021" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
```

```
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."VD_TAB"

ALTER TABLE "BBYAO   "."VD_TAB"
    ADD CONSTRAINT "SQL030417184500211" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."VF_TAB"

ALTER TABLE "BBYAO   "."VF_TAB"
    ADD CONSTRAINT "SQL030417184500391" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."ET_CR_TAB"

ALTER TABLE "BBYAO   "."ET_CR_TAB"
    ADD CONSTRAINT "SQL030417184500541" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."S_TAB"

ALTER TABLE "BBYAO   "."S_TAB"
    ADD CONSTRAINT "SQL030417184500721" FOREIGN KEY
        ("E_ID")
    REFERENCES "BBYAO   "."DICTIONARY_TAB"
        ("E_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."DEF_CR_TAB"

ALTER TABLE "BBYAO   "."DEF_CR_TAB"
```

```
    ADD FOREIGN KEY
        ("E_ID", "S_ID")
    REFERENCES "BBYAO   "."S_TAB"
        ("E_ID", "S_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."Q_TAB"

ALTER TABLE "BBYAO   "."Q_TAB"
    ADD FOREIGN KEY
        ("E_ID", "S_ID")
    REFERENCES "BBYAO   "."S_TAB"
        ("E_ID", "S_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."QT_CR_TAB"

ALTER TABLE "BBYAO   "."QT_CR_TAB"
    ADD FOREIGN KEY
        ("E_ID", "S_ID", "Q_ID")
    REFERENCES "BBYAO   "."Q_TAB"
        ("E_ID", "S_ID", "Q_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."QT_I_TAB"

ALTER TABLE "BBYAO   "."QT_I_TAB"
    ADD FOREIGN KEY
        ("E_ID", "S_ID", "Q_ID")
    REFERENCES "BBYAO   "."Q_TAB"
        ("E_ID", "S_ID", "Q_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."QT_B_TAB"

ALTER TABLE "BBYAO   "."QT_B_TAB"
    ADD FOREIGN KEY
        ("E_ID", "S_ID", "Q_ID")
    REFERENCES "BBYAO   "."Q_TAB"
        ("E_ID", "S_ID", "Q_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
```

```
    ENABLE QUERY OPTIMIZATION;
```

C.2 DB2 DDL for TCMD Class (XML Column)

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."ARTICLE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ARTICLE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "ARTICLE" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."ARTICLE_TAB"

ALTER TABLE "BBYAO   "."ARTICLE_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ARTICLE_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ARTICLE_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "ARTICLE_ID" INTEGER ,
          "TITLE" VARCHAR(300) ,
          "DATE" DATE ,
          "COUNTRY" VARCHAR(50) ,
          "GENRE" VARCHAR(50) )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."AUTHOR_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."AUTHOR_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "NAME" VARCHAR(50) )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."SECTION_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SECTION_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "HEADING" VARCHAR(200) )
         IN "USERSPACE1" ;
```

```
-------------------------------------------------
-- DDL Statements for table "BBYAO   "."P_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."P_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "P" VARCHAR(3000) )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."PHONE_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."PHONE_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "PHONE" VARCHAR(30) )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."EMAIL_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."EMAIL_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "EMAIL" VARCHAR(100) )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."REF_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."REF_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "A_ID" INTEGER )
         IN "USERSPACE1" ;
```

C.3 DB2 DDL for TCMD Class (XML Collection)

```
-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ARTICLE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ARTICLE_TAB"  (
          "LANG" CHAR(2) ,
          "ARTICLE_ID" INTEGER NOT NULL ,
          "TITLE" VARCHAR(252) ,
          "CITY" VARCHAR(25) ,
          "COUNTRY" VARCHAR(25) ,
          "DATE" DATE ,
          "GENRE" VARCHAR(20) )
         IN "USERSPACE1" ;
```

```
-- DDL Statements for primary key on Table "BBYAO
"."ARTICLE_TAB"

ALTER TABLE "BBYAO   "."ARTICLE_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ARTICLE_AUTHOR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ARTICLE_AUTHOR_TAB"  (
          "NAME" VARCHAR(60) NOT NULL ,
          "EMAIL" VARCHAR(50) ,
          "PHONE" VARCHAR(25) ,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."ARTICLE_AUTHOR_TAB"

ALTER TABLE "BBYAO   "."ARTICLE_AUTHOR_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "NAME");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."KEYWORD_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."KEYWORD_TAB"  (
          "KEYWORD" VARCHAR(96) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."KEYWORD_TAB"

ALTER TABLE "BBYAO   "."KEYWORD_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "KEYWORD");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."BODY_P_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."BODY_P_TAB"  (
          "P" VARCHAR(3000) NOT NULL ,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ACK_TAB"
```

```
------------------------------------------------

 CREATE TABLE "BBYAO   "."ACK_TAB"  (
          "PA" VARCHAR(3000) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."REF_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."REF_TAB"  (
          "A_ID" INTEGER NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."REF_TAB"

ALTER TABLE "BBYAO   "."REF_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "A_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."BODY_SEC_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."BODY_SEC_TAB"  (
          "SEC_HEADING" VARCHAR(200) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."BODY_SEC_TAB"

ALTER TABLE "BBYAO   "."BODY_SEC_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "SEC_HEADING");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."SEC_P_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SEC_P_TAB"  (
          "P" VARCHAR(3000) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."SEC_SUBSEC_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SEC_SUBSEC_TAB"  (
```

```
          "SUBSEC_HEADING" VARCHAR(200) NOT NULL ,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."SEC_SUBSEC_TAB"

ALTER TABLE "BBYAO   "."SEC_SUBSEC_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."SUBSEC_P_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SUBSEC_P_TAB"  (
          "P" VARCHAR(3000) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"  (
          "SUBSUBSEC_HEADING" VARCHAR(200) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "BBYAO
"."SUBSEC_SUBSUBSEC_TAB"

ALTER TABLE "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING");

------------------------------------------------
-- DDL Statements for table "BBYAO   "."SUBSUBSEC_P_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."SUBSUBSEC_P_TAB"  (
          "P" VARCHAR(3000) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSUBSEC_HEADING" VARCHAR(200) NOT NULL)
```

```
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO
"."SUBSUBSEC_SUBSUBSUBSEC_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."SUBSUBSEC_SUBSUBSUBSEC_TAB"  (
          "SUBSUBSUBSEC_HEADING" VARCHAR(200) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSUBSEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."SUBSUBSEC_SUBSUBSUBSEC_TAB"

ALTER TABLE "BBYAO   "."SUBSUBSEC_SUBSUBSUBSEC_TAB"
    ADD PRIMARY KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING",
        "SUBSUBSUBSEC_HEADING");


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."SUBSUBSUBSEC_P_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."SUBSUBSUBSEC_P_TAB"  (
          "P" VARCHAR(3000) NOT NULL,
          "ARTICLE_ID" INTEGER NOT NULL,
        "SEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSUBSEC_HEADING" VARCHAR(200) NOT NULL,
        "SUBSUBSUBSEC_HEADING" VARCHAR(200) NOT NULL)
         IN "USERSPACE1" ;


-- DDL Statements for foreign keys on Table "BBYAO
"."ARTICLE_AUTHOR_TAB"

ALTER TABLE "BBYAO   "."ARTICLE_AUTHOR_TAB"
    ADD CONSTRAINT "SQL030329095704760" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."KEYWORD_TAB"
```

```
ALTER TABLE "BBYAO   "."KEYWORD_TAB"
    ADD CONSTRAINT "SQL030329095704810" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."BODY_P_TAB"

ALTER TABLE "BBYAO   "."BODY_P_TAB"
    ADD CONSTRAINT "SQL030329095704850" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."ACK_TAB"

ALTER TABLE "BBYAO   "."ACK_TAB"
    ADD CONSTRAINT "SQL030329095704880" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."REF_TAB"

ALTER TABLE "BBYAO   "."REF_TAB"
    ADD CONSTRAINT "SQL030329095704980" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."BODY_SEC_TAB"

ALTER TABLE "BBYAO   "."BODY_SEC_TAB"
    ADD CONSTRAINT "SQL030329095705070" FOREIGN KEY
        ("ARTICLE_ID")
    REFERENCES "BBYAO   "."ARTICLE_TAB"
        ("ARTICLE_ID")
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO   "."SEC_P_TAB"

ALTER TABLE "BBYAO   "."SEC_P_TAB"
    ADD CONSTRAINT "SQL030329095705120" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING")
    REFERENCES "BBYAO   "."BODY_SEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SEC_SUBSEC_TAB"

ALTER TABLE "BBYAO   "."SEC_SUBSEC_TAB"
    ADD CONSTRAINT "SQL030329095705160" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING")
    REFERENCES "BBYAO   "."BODY_SEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SUBSEC_P_TAB"

ALTER TABLE "BBYAO   "."SUBSEC_P_TAB"
    ADD CONSTRAINT "SQL030329095705230" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING")
    REFERENCES "BBYAO   "."SEC_SUBSEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SUBSEC_SUBSUBSEC_TAB"

ALTER TABLE "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"
```

```
    ADD CONSTRAINT "SQL030329095705290" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING")
    REFERENCES "BBYAO   "."SEC_SUBSEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SUBSUBSEC_P_TAB"

ALTER TABLE "BBYAO   "."SUBSUBSEC_P_TAB"
    ADD CONSTRAINT "SQL030329095705340" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING")
    REFERENCES "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SUBSUBSEC_SUBSUBSUBSEC_TAB"

ALTER TABLE "BBYAO   "."SUBSUBSEC_SUBSUBSUBSEC_TAB"
    ADD CONSTRAINT "SQL030329095705390" FOREIGN KEY
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING")
    REFERENCES "BBYAO   "."SUBSEC_SUBSUBSEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."SUBSUBSUBSEC_P_TAB"

ALTER TABLE "BBYAO   "."SUBSUBSUBSEC_P_TAB"
    ADD CONSTRAINT "SQL030329095705480" FOREIGN KEY
```

```
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING",
        "SUBSUBSUBSEC_HEADING")
    REFERENCES "BBYAO   "."SUBSUBSEC_SUBSUBSUBSEC_TAB"
        ("ARTICLE_ID",
        "SEC_HEADING",
        "SUBSEC_HEADING",
        "SUBSUBSEC_HEADING",
        "SUBSUBSUBSEC_HEADING")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;
```

C.4 DB2 DDL for DCSD Class (XML Collection)

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."CATALOG_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CATALOG_TAB"  (
          "ITEM_ID" VARCHAR(10) NOT NULL ,
          "TITLE" VARCHAR(100) ,
          "DATE_OF_RELEASE" DATE ,
          "PUBLISHER_NAME" VARCHAR(100) ,
          "PUBLISHER_NAME_OF_CITY" VARCHAR(25) ,
          "PUBLISHER_NAME_OF_STATE" VARCHAR(25) ,
          "PUBLISHER_ZIP_CODE" VARCHAR(7) ,
          "PUBLISHER_COUNTRY_NAME" VARCHAR(25) ,
          "PUBLISHER_COUNTRY_EX_RATE" DECIMAL(10,4) ,
          "PUBLISHER_COUNTRY_CURRENCY" VARCHAR(25) ,
          "PUBLISHER_FAX_NUMBER" VARCHAR(25) ,
          "PUBLISHER_PHONE_NUMBER" VARCHAR(25) ,
          "PUBLISHER_WEB_SITE" VARCHAR(50) ,
          "SUBJECT" VARCHAR(20) ,
          "DESCRIPTION" VARCHAR(500) ,
          "THUMBNAIL_DATA" VARCHAR(1) ,
          "IMAGE_DATA" VARCHAR(1) ,
          "SUGGESTED_RETAIL_PRICE_CUR" VARCHAR(25) ,
          "SUGGESTED_RETAIL_PRICE" DECIMAL(8,2) ,
          "COST_CURRENCY" VARCHAR(25) ,
          "COST" DECIMAL(8,2) ,
          "WHEN_IS_AVAILABLE" DATE ,
          "QUANTITY_IN_STOCK" INTEGER ,
          "ISBN" VARCHAR(30) ,
          "NUMBER_OF_PAGES" INTEGER ,
          "TYPE_OF_BOOK" VARCHAR(20) ,
          "LENGTH_UNIT" VARCHAR(10) ,
          "LENGTH" DECIMAL(5,1) ,
          "WIDTH_UNIT" VARCHAR(10) ,
          "WIDTH" DECIMAL(5,1) ,
          "HEIGHT_UNIT" VARCHAR(10) ,
          "HEIGHT" DECIMAL(5,1) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
```

```
"."CATALOG_TAB"

ALTER TABLE "BBYAO   "."CATALOG_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CATALOG_RELATED_ITEM_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CATALOG_RELATED_ITEM_TAB"  (
          "ITEM_ID" VARCHAR(10) NOT NULL ,
          "RELATED_ITEM_ID" VARCHAR(10) NOT NULL )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."CATALOG_RELATED_ITEM_TAB"


ALTER TABLE "BBYAO   "."CATALOG_RELATED_ITEM_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID",
         "RELATED_ITEM_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CATALOG_AUTHOR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CATALOG_AUTHOR_TAB"  (
          "FIRST_NAME" VARCHAR(30) NOT NULL ,
          "MIDDLE_NAME" VARCHAR(30) NOT NULL ,
          "LAST_NAME" VARCHAR(30) NOT NULL ,
          "DATE_OF_BIRTH" DATE ,
          "BIOGRAPHY" VARCHAR(500) ,
          "NAME_OF_CITY" VARCHAR(25) ,
          "NAME_OF_STATE" VARCHAR(25) ,
          "ZIP_CODE" VARCHAR(7) ,
          "NAME_OF_COUNTRY" VARCHAR(25) ,
          "PHONE_NUMBER" VARCHAR(25) ,
          "EMAIL_ADDRESS" VARCHAR(50) ,
          "ITEM_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."CATALOG_AUTHOR_TAB"

ALTER TABLE "BBYAO   "."CATALOG_AUTHOR_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID",
         "FIRST_NAME",
         "MIDDLE_NAME",
         "LAST_NAME");


------------------------------------------------
-- DDL Statements for table "BBYAO
"."CATALOG_AUTHOR_ADDRESS_TAB"
```

```
------------------------------------------------

 CREATE TABLE "BBYAO   "."CATALOG_AUTHOR_ADDRESS_TAB"  (
          "STREET_ADDRESS" VARCHAR(40) NOT NULL ,
          "FIRST_NAME" VARCHAR(30) NOT NULL,
          "MIDDLE_NAME" VARCHAR(30) NOT NULL,
          "LAST_NAME" VARCHAR(30) NOT NULL,
          "ITEM_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."CATALOG_AUTHOR_ADDRESS_TAB"

ALTER TABLE "BBYAO   "."CATALOG_AUTHOR_ADDRESS_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID",
        "FIRST_NAME",
        "MIDDLE_NAME",
        "LAST_NAME",
        "STREET_ADDRESS");


------------------------------------------------
-- DDL Statements for table "BBYAO
"."CATALOG_PUBLISHER_ADDRESS_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CATALOG_PUBLISHER_ADDRESS_TAB"  (
          "STREET_ADDRESS" VARCHAR(40) NOT NULL ,
          "ITEM_ID" VARCHAR(10) NOT NULL)
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "BBYAO
"."CATALOG_PUBLISHER_ADDRESS_TAB"

ALTER TABLE "BBYAO   "."CATALOG_PUBLISHER_ADDRESS_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID",
        "STREET_ADDRESS");


-- DDL Statements for foreign keys on Table "BBYAO
"."CATALOG_AUTHOR_TAB"

ALTER TABLE "BBYAO   "."CATALOG_AUTHOR_TAB"
    ADD CONSTRAINT "SQL030421012545311" FOREIGN KEY
        ("ITEM_ID")
    REFERENCES "BBYAO   "."CATALOG_TAB"
        ("ITEM_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."CATALOG_AUTHOR_ADDRESS_TAB"
```

```
ALTER TABLE "BBYAO   "."CATALOG_AUTHOR_ADDRESS_TAB"
    ADD CONSTRAINT "SQL030421012545481" FOREIGN KEY
        ("ITEM_ID",
         "FIRST_NAME",
         "MIDDLE_NAME",
         "LAST_NAME")
    REFERENCES "BBYAO   "."CATALOG_AUTHOR_TAB"
        ("ITEM_ID",
         "FIRST_NAME",
         "MIDDLE_NAME",
         "LAST_NAME")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."CATALOG_PUBLISHER_ADDRESS_TAB"

ALTER TABLE "BBYAO   "."CATALOG_PUBLISHER_ADDRESS_TAB"
    ADD CONSTRAINT "SQL030421012545651" FOREIGN KEY
        ("ITEM_ID")
    REFERENCES "BBYAO   "."CATALOG_TAB"
        ("ITEM_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;


-- DDL Statements for foreign keys on Table "BBYAO
"."CATALOG_RELATED_ITEM_TAB"

ALTER TABLE "BBYAO   "."CATALOG_RELATED_ITEM_TAB"
    ADD CONSTRAINT "SQL030421012545783" FOREIGN KEY
        ("ITEM_ID")
    REFERENCES "BBYAO   "."CATALOG_TAB"
        ("ITEM_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;
```

C.5 DB2 DDL for DCMD Class (XML Column)

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."ADDRESS_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ADDRESS_TAB"  (
           "ID" INTEGER NOT NULL ,
           "ADDRESS" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
          IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."ADDRESS_TAB"
```

```
ALTER TABLE "BBYAO   "."ADDRESS_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."AUTHOR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."AUTHOR_TAB"  (
          "ID" INTEGER NOT NULL ,
          "AUTHOR" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."AUTHOR_TAB"

ALTER TABLE "BBYAO   "."AUTHOR_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."COUNTRY_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."COUNTRY_TAB"  (
          "ID" INTEGER NOT NULL ,
          "COUNTRY" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."COUNTRY_TAB"

ALTER TABLE "BBYAO   "."COUNTRY_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."AUTHOR_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."AUTHOR_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "AUTHOR_ID" INTEGER )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."AUTHOR_BIO_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."AUTHOR_BIO_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "BIOGRAPHY" VARCHAR(500) )
         IN "USERSPACE1" ;
```

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."ITEM_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ITEM_TAB"  (
          "ID" INTEGER NOT NULL ,
          "ITEM" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."ITEM_TAB"

ALTER TABLE "BBYAO   "."ITEM_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ITEM_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ITEM_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "ITEM_ID" INTEGER )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_TAB"  (
          "ID" INTEGER NOT NULL ,
          "ORDER" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."ORDER_TAB"

ALTER TABLE "BBYAO   "."ORDER_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "ORDER_ID" INTEGER ,
          "CUSTOMER_ID" INTEGER ,
          "TOTAL" DECIMAL(10,2) ,
          "ORDER_DATE" DATE ,
          "SHIP_TYPE" VARCHAR(10) ,
          "ORDER_STATUS" VARCHAR(10) )
         IN "USERSPACE1" ;
```

```
-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "ORDER_LINE_ID" INTEGER )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_DISC_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_DISC_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "DISCOUNT_RATE" DECIMAL(5,2) )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_ITEM_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_ITEM_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "ITEM_ID" INTEGER )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_QUAN_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_QUAN_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "QUANTITY_OF_ITEM" INTEGER )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_INSTR_SIDE_TAB"
-------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_INSTR_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "SPECIAL_INSTRUCTIONS" VARCHAR(100) )
         IN "USERSPACE1" ;


-------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_TAB"
-------------------------------------------------
```

```
 CREATE TABLE "BBYAO   "."CUSTOMER_TAB"  (
          "ID" INTEGER NOT NULL ,
          "CUSTOMER" "DB2XML  "."XMLCLOB" NOT LOGGED NOT COMPACT )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."CUSTOMER_TAB"

ALTER TABLE "BBYAO   "."CUSTOMER_TAB"
    ADD PRIMARY KEY
        ("ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CUSTOMER_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "CUSTOMER_ID" INTEGER )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_FNAME_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CUSTOMER_FNAME_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "FIRST_NAME" VARCHAR(50) )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_LNAME_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CUSTOMER_LNAME_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "LAST_NAME" VARCHAR(50) )
         IN "USERSPACE1" ;


------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_PHONE_SIDE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CUSTOMER_PHONE_SIDE_TAB"  (
          "ID" INTEGER NOT NULL ,
          "DXX_SEQNO" INTEGER ,
          "PHONE_NUMBER" VARCHAR(30) )
         IN "USERSPACE1" ;
```

C.6 DB2 DDL for DCMD Class (XML Collection)

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."CUSTOMER_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."CUSTOMER_TAB"  (
          "CUSTOMER_ID" INTEGER NOT NULL ,
          "USER_NAME" VARCHAR(20) ,
          "PASSWORD" VARCHAR(20) ,
          "FIRST_NAME" VARCHAR(15) ,
          "LAST_NAME" VARCHAR(15) ,
          "ADDRESS_ID" INTEGER ,
          "PHONE_NUMBER" VARCHAR(16) ,
          "EMAIL_ADDRESS" VARCHAR(50) ,
          "DATE_OF_REGISTRATION" DATE ,
          "DATE_OF_LAST_VISIT" DATE ,
          "START_OF_CURRENT_SESSION" CHAR(29) ,
          "CURRENT_SESSION_EXPIRY" CHAR(29) ,
          "DISCOUNT_RATE" DECIMAL(3,2) ,
          "BALANCE" DECIMAL(3,2) ,
          "YTD_PAYMENT" DECIMAL(5,2) ,
          "BIRTH_DATE" DATE ,
          "MISCELLANEOUS_INFORMATION" VARCHAR(500) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."CUSTOMER_TAB"

ALTER TABLE "BBYAO   "."CUSTOMER_TAB"
    ADD PRIMARY KEY
        ("CUSTOMER_ID");



------------------------------------------------
-- DDL Statements for table "BBYAO   "."ITEM_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ITEM_TAB"  (
          "ITEM_ID" INTEGER NOT NULL ,
          "TITLE" VARCHAR(80) ,
          "AUTHOR_ID" INTEGER ,
          "DATE_OF_RELEASE" DATE ,
          "NAME_OF_PUBLISHER" VARCHAR(60) ,
          "SUBJECT" VARCHAR(15) ,
          "DESCRIPTION" VARCHAR(500) ,
          "THUMBNAIL" VARCHAR(1) ,
          "IMAGE" VARCHAR(1) ,
          "SUGGESTED_RETAIL_PRICE" DECIMAL(6,2) ,
          "COST" DECIMAL(6,2) ,
          "WHEN_IS_AVAILABLE" DATE ,
          "QUANTITY_IN_STOCK" INTEGER ,
          "ISBN" CHAR(14) ,
          "NUMBER_OF_PAGES" INTEGER ,
          "TYPE_OF_BOOK" VARCHAR(15) ,
          "SIZE_OF_BOOK" VARCHAR(17) )
         IN "USERSPACE1" ;
```

```
-- DDL Statements for primary key on Table "BBYAO   "."ITEM_TAB"

ALTER TABLE "BBYAO   "."ITEM_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID");



------------------------------------------------
-- DDL Statements for table "BBYAO   "."RELATED_ITEM_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."RELATED_ITEM_TAB"  (
          "RELATED_ITEM_ID" INTEGER NOT NULL,
          "ITEM_ID" INTEGER NOT NULL )
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "BBYAO
"."RELATED_ITEM_TAB"

ALTER TABLE "BBYAO   "."RELATED_ITEM_TAB"
    ADD PRIMARY KEY
        ("ITEM_ID",
        "RELATED_ITEM_ID");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_TAB"  (
          "ORDER_ID" INTEGER NOT NULL ,
          "CUSTOMER_ID" INTEGER ,
          "ORDER_DATE" DATE ,
          "SUBTOTAL" DECIMAL(6,2) ,
          "TAXL" DECIMAL(5,2) ,
          "TOTAL" DECIMAL(7,2) ,
          "SHIP_TYPE" VARCHAR(10) ,
          "SHIP_DATE" DATE ,
          "BILL_ADDRESS_ID" INTEGER ,
          "SHIP_ADDRESS_ID" INTEGER ,
          "ORDER_STATUS" VARCHAR(10) ,
          "CREDIT_CARD_TYPE" VARCHAR(10) ,
          "CREDIT_CARD_NUMBER" VARCHAR(16) ,
          "NAME_ON_CREDIT_CARD" VARCHAR(30) ,
          "EXPIRATION_DATE" DATE ,
          "AUTHORIZATION_ID" VARCHAR(15) ,
          "TRANSACTION_AMOUNT" DECIMAL(7,2) ,
          "AUTHORIZATION_DATE" DATE ,
          "TRANSACTION_COUNTRY_ID" INTEGER )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."ORDER_TAB"

ALTER TABLE "BBYAO   "."ORDER_TAB"
    ADD PRIMARY KEY
        ("ORDER_ID");
```

```
------------------------------------------------
-- DDL Statements for table "BBYAO   "."ORDER_LINE_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ORDER_LINE_TAB"  (
          "ORDER_LINE_ID" INTEGER NOT NULL,
          "ITEM_ID" INTEGER ,
          "QUANTITY_OF_ITEM" INTEGER ,
          "DISCOUNT_RATE" DECIMAL(3,2) ,
          "SPECIAL_INSTRUCTIONS" VARCHAR(100) ,
          "ORDER_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "BBYAO
"."ORDER_LINE_TAB"

ALTER TABLE "BBYAO   "."ORDER_LINE_TAB"
    ADD PRIMARY KEY
        ("ORDER_ID",
        "ORDER_LINE_ID");



------------------------------------------------
-- DDL Statements for table "BBYAO   "."ADDRESS_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."ADDRESS_TAB"  (
          "ADDRESS_ID" INTEGER NOT NULL ,
          "NAME_OF_CITY" VARCHAR(30) ,
          "NAME_OF_STATE" VARCHAR(20) ,
          "ZIP_CODE" VARCHAR(10) ,
          "COUNTRY_ID" INTEGER )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."ADDRESS_TAB"

ALTER TABLE "BBYAO   "."ADDRESS_TAB"
    ADD PRIMARY KEY
        ("ADDRESS_ID");



------------------------------------------------
-- DDL Statements for table "BBYAO   "."STREET_ADDRESS_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."STREET_ADDRESS_TAB"  (
          "STREET_ADDRESS" VARCHAR(40) NOT NULL,
          "ADDRESS_ID" INTEGER NOT NULL)
         IN "USERSPACE1" ;


-- DDL Statements for primary key on Table "BBYAO
"."STREET_ADDRESS_TAB"
```

```
ALTER TABLE "BBYAO   "."STREET_ADDRESS_TAB"
    ADD PRIMARY KEY
        ("ADDRESS_ID",
        "STREET_ADDRESS");


------------------------------------------------
-- DDL Statements for table "BBYAO   "."AUTHOR_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."AUTHOR_TAB"  (
          "AUTHOR_ID" INTEGER NOT NULL ,
          "FIRST_NAME" VARCHAR(20) ,
          "MIDDLE_NAME" VARCHAR(20) ,
          "LAST_NAME" VARCHAR(30) ,
          "DATE_OF_BIRTH" DATE ,
          "BIOGRAPHY" VARCHAR(500) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO   "."AUTHOR_TAB"

ALTER TABLE "BBYAO   "."AUTHOR_TAB"
    ADD PRIMARY KEY
        ("AUTHOR_ID");



------------------------------------------------
-- DDL Statements for table "BBYAO   "."COUNTRY_TAB"
------------------------------------------------

 CREATE TABLE "BBYAO   "."COUNTRY_TAB"  (
          "COUNTRY_ID" INTEGER NOT NULL ,
          "NAME" VARCHAR(20) ,
          "EXCHANGE_RATE" DECIMAL(12,6) ,
          "CURRENCY" VARCHAR(20) )
         IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "BBYAO
"."COUNTRY_TAB"

ALTER TABLE "BBYAO   "."COUNTRY_TAB"
    ADD PRIMARY KEY
        ("COUNTRY_ID");




-- DDL Statements for foreign keys on Table "BBYAO
"."RELATED_ITEM_TAB"

ALTER TABLE "BBYAO   "."RELATED_ITEM_TAB"
    ADD CONSTRAINT "SQL030329052533270" FOREIGN KEY
        ("ITEM_ID")
    REFERENCES "BBYAO   "."ITEM_TAB"
        ("ITEM_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
```

```
    ENFORCED
    ENABLE QUERY OPTIMIZATION;



-- DDL Statements for foreign keys on Table "BBYAO
"."ORDER_LINE_TAB"

ALTER TABLE "BBYAO   "."ORDER_LINE_TAB"
    ADD CONSTRAINT "SQL030329060742450" FOREIGN KEY
        ("ORDER_ID")
    REFERENCES "BBYAO   "."ORDER_TAB"
        ("ORDER_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;



-- DDL Statements for foreign keys on Table "BBYAO
"."STREET_ADDRESS_TAB"

ALTER TABLE "BBYAO   "."STREET_ADDRESS_TAB"
    ADD CONSTRAINT "SQL030329064355500" FOREIGN KEY
        ("ADDRESS_ID")
    REFERENCES "BBYAO   "."ADDRESS_TAB"
        ("ADDRESS_ID")
    ON DELETE NO ACTION
    ON UPDATE NO ACTION
    ENFORCED
    ENABLE QUERY OPTIMIZATION;
```

Appendix D

SQL Server Annotated XSDs

D.1 Annotated XSD for Dictionary

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com)
<!--  by Benjamin Bin Yao (University of Waterloo) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="hw_details" parent="dictionary_tab"
        parent-key="entry_id" child="hw_tab" child-key="entry_id"/>
      <sql:relationship name="pr_details" parent="dictionary_tab"
        parent-key="entry_id" child="pr_tab" child-key="entry_id"/>
      <sql:relationship name="pos_details" parent="dictionary_tab"
        parent-key="entry_id" child="pos_tab" child-key="entry_id"/>
      <sql:relationship name="vd_details" parent="dictionary_tab"
        parent-key="entry_id" child="vd_tab" child-key="entry_id"/>
      <sql:relationship name="vf_details" parent="dictionary_tab"
        parent-key="entry_id" child="vf_tab" child-key="entry_id"/>
      <sql:relationship name="et_cr_details" parent="dictionary_tab"
        parent-key="entry_id" child="et_cr_tab" child-key="entry_id"/>
      <sql:relationship name="s_details" parent="dictionary_tab"
        parent-key="entry_id" child="s_tab" child-key="entry_id"/>
      <sql:relationship name="def_cr_details" parent="s_tab"
        parent-key="s_id" child="def_cr_tab" child-key="s_id"/>
      <sql:relationship name="q_details" parent="s_tab"
        parent-key="s_id" child="q_tab" child-key="s_id"/>
      <sql:relationship name="qt_cr_details" parent="q_tab"
        parent-key="q_id" child="qt_cr_tab" child-key="q_id"/>
      <sql:relationship name="qt_i_details" parent="q_tab"
        parent-key="q_id" child="qt_i_tab" child-key="q_id"/>
      <sql:relationship name="qt_b_details" parent="q_tab"
        parent-key="q_id" child="qt_b_tab" child-key="q_id"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:element name="a" type="xsd:string"/>
  <xsd:element name="bib">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
```

```
</xsd:element>
<xsd:element name="def" sql:is-constant="1">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="4">
      <xsd:element name="cr" type="xsd:string" sql:relation="def_cr_tab"
        sql:relationship="s_details def_cr_details"
        sql:key-fields="entry_id s_id cr"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="dictionary" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="e" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="e" sql:relation="dictionary_tab" sql:key-fields="entry_id">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="hwg"/>
      <xsd:element ref="vfl" minOccurs="0"/>
      <xsd:element ref="et" minOccurs="0"/>
      <xsd:element ref="ss"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" sql:field="entry_id"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="et" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence maxOccurs="16">
      <xsd:element name="cr" type="xsd:string" sql:relation="et_cr_tab"
        sql:relationship="et_cr_details" sql:key-fields="entry_id cr"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="hw" sql:relation="hw_tab" sql:relationship="hw_details"
    sql:key-fields="entry_id hw">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="hwg" sql:is-constant="1">
  <xsd:complexType>
    <xsd:choice maxOccurs="15">
      <xsd:element ref="hw"/>
      <xsd:element ref="pr" minOccurs="0"/>
      <xsd:element ref="pos" minOccurs="0"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="pos" type="xsd:string" sql:relation="pos_tab"
  sql:relationship="pos_details" sql:key-fields="entry_id pos"/>
<xsd:element name="pr" type="xsd:string" sql:relation="pr_tab"
  sql:relationship="pr_details" sql:key-fields="entry_id pr"/>
<xsd:element name="q" sql:relation="q_tab" sql:relationship="s_details q_details"
  sql:key-fields="entry_id s_id q_id">
  <xsd:complexType>
    <xsd:sequence>
```

```
      <xsd:element ref="qd"/>
      <xsd:element ref="a" minOccurs="0"/>
      <xsd:element ref="w"/>
      <xsd:element ref="bib" minOccurs="0"/>
      <xsd:element ref="loc"/>
      <xsd:element ref="qt"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" sql:field="q_id"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="qd">
  <xsd:simpleType>
    <xsd:restriction base="xsd:short"/>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="qt" sql:is-constant="1">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0" maxOccurs="6">
      <xsd:element name="cr" type="xsd:string" sql:relation="qt_cr_tab"
          sql:relationship="s_details q_details qt_cr_details"
          sql:key-fields="entry_id s_id q_id cr"/>
      <xsd:element name="i" type="xsd:string" sql:relation="qt_i_tab"
          sql:relationship="s_details q_details qt_i_details"
          sql:key-fields="entry_id s_id q_id i"/>
      <xsd:element name="b" type="xsd:string" sql:relation="qt_b_tab"
          sql:relationship="s_details q_details qt_b_details"
          sql:key-fields="entry_id s_id q_id b"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
<xsd:element name="loc" type="xsd:string"/>
<xsd:element name="qp" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="q" maxOccurs="20"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="s" sql:relation="s_tab" sql:relationship="s_details"
  sql:key-fields="entry_id s_id">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="def"/>
      <xsd:element ref="qp"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" sql:field="s_id"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="ss" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="s" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="vd" sql:relation="vd_tab" sql:relationship="vd_details"
  sql:key-fields="entry_id vd">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string"/>
```

```
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="vf" sql:relation="vf_tab" sql:relationship="vf_details"
    sql:key-fields="entry_id vf">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="vfl" sql:is-constant="1">
    <xsd:complexType>
      <xsd:choice maxOccurs="45">
        <xsd:element ref="vd" minOccurs="0"/>
        <xsd:element ref="vf"/>
      </xsd:choice>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="w" type="xsd:string"/>
</xsd:schema>
```

D.2 Annotated XSD for Articles

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="author_details" parent="article_tab"
        parent-key="article_id" child="article_author_tab"
        child-key="article_id"/>
      <sql:relationship name="keyword_details" parent="article_tab"
        parent-key="article_id" child="article_keyword_tab"
        child-key="article_id"/>
      <sql:relationship name="ack_details" parent="article_tab"
        parent-key="article_id" child="article_ack_tab" child-key="article_id"/>
      <sql:relationship name="ref_details" parent="article_tab"
        parent-key="article_id" child="article_ref_tab" child-key="article_id"/>
      <sql:relationship name="body_p_details" parent="article_tab"
        parent-key="article_id" child="body_p_tab" child-key="article_id"/>
      <sql:relationship name="body_sec_details" parent="article_tab"
        parent-key="article_id" child="body_sec_tab" child-key="article_id"/>
      <sql:relationship name="sec_p_details" parent="body_sec_tab"
        parent-key="sec_heading" child="sec_p_tab" child-key="sec_heading"/>
      <sql:relationship name="sec_subsec_details" parent="body_sec_tab"
        parent-key="sec_heading" child="sec_subsec_tab" child-key="sec_heading"/>
      <sql:relationship name="subsec_p_details" parent="sec_subsec_tab"
        parent-key="subsec_heading" child="subsec_p_tab"
        child-key="subsec_heading"/>
      <sql:relationship name="subsec_subsubsec_details" parent="sec_subsec_tab"
        parent-key="subsec_heading" child="subsec_subsubsec_tab"
        child-key="subsec_heading"/>
      <sql:relationship name="subsubsec_p_details" parent="subsec_subsubsec_tab"
        parent-key="subsubsec_heading" child="subsubsec_p_tab"
        child-key="subsubsec_heading"/>
      <sql:relationship name="subsubsec_subsubsubsec_details"
        parent="subsec_subsubsec_tab" parent-key="subsubsec_heading"
        child="subsubsec_subsubsubsec_tab" child-key="subsubsec_heading"/>
      <sql:relationship name="subsubsubsec_p_details"
        parent="subsubsec_subsubsubsec_tab"
```

```
      parent-key="subsubsubsec_heading" child="subsubsubsec_p_tab"
      child-key="subsubsubsec_heading"/>
  </xsd:appinfo>
</xsd:annotation>
<xsd:element name="a_id" type="xsd:string" sql:relation="article_ref_tab"
  sql:relationship="ref_details" sql:key-fields="articler_id a_id"/>
<xsd:element name="abstract" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="p" type="xsd:string" sql:relation="body_p_tab"
        sql:relationship="body_p_details" sql:key-fields="article_id p"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="article" sql:relation="article_tab"
  sql:key-fields="article_id">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="prolog"/>
      <xsd:element ref="body"/>
      <xsd:element name="epilog" sql:is-constant="1">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="acknowledgements" minOccurs="0"
              sql:is-constant="1">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element ref="pa" maxOccurs="3"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="references" minOccurs="0"
              sql:is-constant="1">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element ref="a_id" maxOccurs="unbounded"/>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:byte" use="required"
      sql:field="article_id"/>
    <xsd:attribute name="lang" type="xsd:string" use="required"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="author" sql:relation="article_author_tab"
  sql:relationship="author_details" sql:key-fields="articler_id name">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element ref="contact"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="authors" sql:is-constant="1">
  <xsd:complexType>
```

```
    <xsd:sequence>
      <xsd:element ref="author" maxOccurs="48"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="body" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="abstract" minOccurs="0"/>
      <xsd:element ref="section" maxOccurs="15"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="section"  sql:relation="body_sec_tab"
  sql:relationship="body_sec_details" sql:key-fields="article_id sec_heading">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="p" type="xsd:string" sql:relation="sec_p_tab"
        sql:relationship="body_sec_details sec_p_details"
        sql:key-fields="sec_heading p"/>
      <xsd:element ref="subsec" minOccurs="0" maxOccurs="23"/>
    </xsd:sequence>
    <xsd:attribute name="heading" type="xsd:string" use="required"
      sql:field="sec_heading"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="city" type="xsd:string"/>
<xsd:element name="contact" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="email" minOccurs="0"/>
      <xsd:element ref="phone" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="country" type="xsd:string"/>
<xsd:element name="date" type="xsd:date"/>
<xsd:element name="dateline" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="city"/>
      <xsd:element ref="country"/>
      <xsd:element ref="date"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="email">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="genre" type="xsd:string"/>
<xsd:element name="keyword" type="xsd:string"/>
<xsd:element name="keywords" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="keyword" sql:relation="article_keyword_tab"
        sql:relationship="keyword_details" sql:key-fields="articler_id keyword"/>
    </xsd:sequence>
```

```
  </xsd:complexType>
</xsd:element>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="pa" type="xsd:string" sql:relation="article_ack_tab"
  sql:relationship="ack_details" sql:key-fields="articler_id name"/>
<xsd:element name="phone" type="xsd:string"/>
<xsd:element name="prolog" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="title"/>
      <xsd:element ref="authors" minOccurs="0"/>
      <xsd:element ref="dateline" minOccurs="0"/>
      <xsd:element ref="genre" minOccurs="0"/>
      <xsd:element ref="keywords" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="subsec" sql:relation="sec_subsec_tab"
  sql:relationship="body_sec_details sec_subsec_details"
  sql:key-fields="sec_heading">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="p" type="xsd:string" sql:relation="subsec_p_tab"
        sql:relationship="body_sec_details sec_p_details subsec_p_details"
        sql:key-fields="subsec_heading p"/>
      <xsd:element name="subsec" sql:relation="subsec_subsubsec_tab"
        sql:relationship="body_sec_details sec_subsec_details
          subsec_subsubsec_details"
        sql:key-fields="subsec_heading">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="p" type="xsd:string"
              sql:relation="subsubsec_p_tab"
              sql:relationship="body_sec_details sec_p_details subsec_p_details
              subsubsec_p_details" sql:key-fields="subsubsec_heading p"/>
            <xsd:element name="subsec" sql:relation="subsubsec_subsubsubsec_tab"
              sql:relationship="body_sec_details sec_subsec_details
              subsec_subsubsec_details subsubsec_subsubsubsec_details"
              sql:key-fields="subsubsec_heading">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="p" type="xsd:string"
                    sql:relation="subsubsubsec_p_tab"
                    sql:relationship="body_sec_details sec_p_details
                    subsec_p_details
                    subsubsec_p_details subsubsubsec_p_details"
                    sql:key-fields="subsubsubsec_heading p"/>
                </xsd:sequence>
                <xsd:attribute name="heading" type="xsd:string" use="required"
                  sql:field="subsubsubsec_heading"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
          <xsd:attribute name="heading" type="xsd:string" use="required"
            sql:field="subsubsec_heading"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="heading" type="xsd:string" use="required"
```

```
        sql:field="subsec_heading"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="title" type="xsd:string"/>
</xsd:schema>
```

D.3 Annotated XSD for Catalog

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="author_details" parent="catalog_tab"
        parent-key="item_id"
        child="catalog_author_tab" child-key="item_id"/>
      <sql:relationship name="author_address_details"
        parent="catalog_author_tab"
        parent-key="first_name middle_name last_name"
        child="catalog_author_address_tab"
        child-key="first_name middle_name last_name"/>
      <sql:relationship name="related_item_details"
        parent="catalog_tab"
        parent-key="item_id" child="catalog_related_item_tab"
        child-key="item_id"/>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:element name="FAX_number" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="attributes" sql:is-constant="1">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="ISBN"/>
        <xsd:element ref="number_of_pages"/>
        <xsd:element ref="type_of_book"/>
        <xsd:element ref="size_of_book"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="author" sql:relation="catalog_author_tab"
    sql:relationship="author_details"
    sql:key-fields="item_id first_name middle_name last_name">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="name" sql:is-constant="1">
          <xsd:complexType>
            <xsd:all>
              <xsd:element ref="first_name"/>
              <xsd:element ref="middle_name"/>
              <xsd:element ref="last_name"/>
            </xsd:all>
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="date_of_birth"/>
        <xsd:element ref="biography"/>
        <xsd:element name="contact_information" sql:is-constant="1">
          <xsd:complexType>
            <xsd:all>
```

```
            <xsd:element name="mailing_address" sql:is-constant="1">
              <xsd:complexType>
                <xsd:all>
                  <xsd:element ref="street_information"/>
                  <xsd:element ref="name_of_city"/>
                  <xsd:element ref="name_of_state"/>
                  <xsd:element ref="zip_code"/>
                  <xsd:element name="name_of_country" type="xsd:string"/>
                </xsd:all>
              </xsd:complexType>
            </xsd:element>
            <xsd:element ref="phone_number"/>
            <xsd:element ref="email_address"/>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="authors" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="author" maxOccurs="4"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="biography" type="xsd:string"/>
<xsd:element name="catalog" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="item" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="cost">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="country" sql:is-constant="1">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element ref="exchange_rate"/>
      <xsd:element ref="currency"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="currency" type="xsd:string"/>
<xsd:element name="data" type="xsd:string"/>
<xsd:element name="date_of_birth" type="xsd:dateTime"/>
<xsd:element name="date_of_release" type="xsd:dateTime"/>
<xsd:element name="description" type="xsd:string"/>
<xsd:element name="email_address" type="xsd:string"/>
<xsd:element name="web_site" type="xsd:string"/>
```

```
<xsd:element name="exchange_rate" type="xsd:decimal"/>
<xsd:element name="first_name" type="xsd:string"/>
<xsd:element name="height">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="unit" type="xsd:string" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="image">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="data"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="item" sql:relation="catalog_tab" sql:key-fields="item_id">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="title"/>
      <xsd:element ref="authors"/>
      <xsd:element ref="date_of_release"/>
      <xsd:element ref="publisher"/>
      <xsd:element ref="subject"/>
      <xsd:element ref="description"/>
      <xsd:element ref="related_items"/>
      <xsd:element ref="media"/>
      <xsd:element ref="pricing"/>
      <xsd:element ref="attributes"/>
    </xsd:all>
    <xsd:attribute name="id" type="xsd:ID" sql:field="item_id"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="item_id" type="xsd:string" sql:field="related_item_id"/>
<xsd:element name="last_name" type="xsd:string"/>
<xsd:element name="length">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="unit" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="media" sql:is-constant="1">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="thumbnail"/>
      <xsd:element ref="image"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="middle_name" type="xsd:string"/>
<xsd:element name="name_of_city" type="xsd:string"/>
<xsd:element name="name_of_country" type="xsd:string"/>
<xsd:element name="name_of_state" type="xsd:string"/>
<xsd:element name="number_of_pages" type="xsd:short"/>
```

```
<xsd:element name="phone_number" type="xsd:string"/>
<xsd:element name="pricing" sql:is-constant="1">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="suggested_retail_price"/>
      <xsd:element ref="cost"/>
      <xsd:element ref="when_is_available"/>
      <xsd:element ref="quantity_in_stock"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="publisher" sql:is-constant="1">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="contact_information" sql:is-constant="1">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="mailing_address" sql:is-constant="1">
              <xsd:complexType>
                <xsd:all>
                  <xsd:element ref="street_information"/>
                  <xsd:element ref="name_of_city"/>
                  <xsd:element ref="name_of_state"/>
                  <xsd:element ref="zip_code"/>
                  <xsd:element ref="country"/>
                </xsd:all>
              </xsd:complexType>
            </xsd:element>
            <xsd:element ref="FAX_number" minOccurs="0"/>
            <xsd:element ref="phone_number"/>
            <xsd:element ref="web_site"/>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="quantity_in_stock" type="xsd:byte"/>
<xsd:element name="related_item" sql:relation="catalog_related_item_tab"
  sql:relationship="related_item_details"
  sql:key-fields="item_id related_item_id">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="item_id"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
<xsd:element name="related_items" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="related_item" minOccurs="0" maxOccurs="5"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="size_of_book" sql:is-constant="1">
  <xsd:complexType>
    <xsd:all>
      <xsd:element ref="length"/>
```

```
        <xsd:element ref="width"/>
        <xsd:element ref="height"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="street_address" type="xsd:string"
    sql:relation="catalog_author_address_tab"
    sql:relationship="author_details author_address_details"
    sql:key-fields="first_name middle_name last_name"/>
  <xsd:element name="street_information" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="street_address" maxOccurs="2"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="subject" type="xsd:string"/>
  <xsd:element name="suggested_retail_price">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:decimal">
          <xsd:attribute name="currency" type="xsd:string" use="required"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="thumbnail">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="data"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="title" type="xsd:string"/>
  <xsd:element name="type_of_book" type="xsd:string"/>
  <xsd:element name="when_is_available" type="xsd:dateTime"/>
  <xsd:element name="width">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:decimal">
          <xsd:attribute name="unit" type="xsd:string" use="required"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="zip_code" type="xsd:string"/>
</xsd:schema>
```

D.4 Annotated XSD for Item

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="related_item_details" parent="item_tab"
      parent-key="item_id" child="related_item_id_tab" child-key="item_id"/>
```

```
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="author_id" type="xsd:integer"/>
  <xsd:element name="cost" type="xsd:decimal"/>
  <xsd:element name="date_of_release" type="xsd:dateTime"/>
  <xsd:element name="description" type="xsd:string"/>
  <xsd:element name="image" type="xsd:string"/>
  <xsd:element name="item" sql:relation="item_tab" sql:key-fields="item_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="title"/>
        <xsd:element ref="author_id"/>
        <xsd:element ref="date_of_release"/>
        <xsd:element ref="name_of_publisher"/>
        <xsd:element ref="subject"/>
        <xsd:element ref="description"/>
        <xsd:element ref="related_item_id"/>
        <xsd:element ref="thumbnail"/>
        <xsd:element ref="image"/>
        <xsd:element ref="suggested_retail_price"/>
        <xsd:element ref="cost"/>
        <xsd:element ref="when_is_available"/>
        <xsd:element ref="quantity_in_stock"/>
        <xsd:element ref="ISBN"/>
        <xsd:element ref="number_of_pages"/>
        <xsd:element ref="type_of_book"/>
        <xsd:element ref="size_of_book"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="item_id"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="items" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="item"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="name_of_publisher" type="xsd:string"/>
  <xsd:element name="number_of_pages" type="xsd:short"/>
  <xsd:element name="quantity_in_stock" type="xsd:byte"/>
  <xsd:element name="related_item_id" type="xsd:integer"
    sql:relation="related_item_id_tab" sql:relationship="related_item_details"
    sql:key-fields="item_id related_item_id"/>
  <xsd:element name="size_of_book" type="xsd:string"/>
  <xsd:element name="subject" type="xsd:string"/>
  <xsd:element name="suggested_retail_price" type="xsd:decimal"/>
  <xsd:element name="thumbnail" type="xsd:string"/>
  <xsd:element name="title" type="xsd:string"/>
  <xsd:element name="type_of_book" type="xsd:string"/>
  <xsd:element name="when_is_available" type="xsd:dateTime"/>
</xsd:schema>
```

D.5 Annotated XSD for Order

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
<xsd:annotation>
  <xsd:appinfo>
    <sql:relationship name="order_details" parent="order_tab"
      parent-key="order_id"
      child="order_line_tab" child-key="order_id"/>
  </xsd:appinfo>
</xsd:annotation>
<xsd:element name="order" sql:relation="order_tab" sql:key-fields="order_id">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="customer_id"/>
      <xsd:element ref="order_date"/>
      <xsd:element ref="subtotal"/>
      <xsd:element ref="tax"/>
      <xsd:element ref="total"/>
      <xsd:element ref="ship_type"/>
      <xsd:element ref="ship_date"/>
      <xsd:element ref="bill_address_id"/>
      <xsd:element ref="ship_address_id"/>
      <xsd:element ref="order_status"/>
      <xsd:element ref="credit_card_transaction"/>
      <xsd:element ref="order_lines"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:integer" sql:field="order_id"/>
  </xsd:complexType>
</xsd:element>
<xsd:element name="customer_id" type="xsd:integer"/>
<xsd:element name="order_date" type="xsd:dateTime"/>
<xsd:element name="subtotal" type="xsd:decimal"/>
<xsd:element name="tax" type="xsd:decimal"/>
<xsd:element name="total" type="xsd:decimal"/>
<xsd:element name="ship_type" type="xsd:string"/>
<xsd:element name="ship_date" type="xsd:dateTime"/>
<xsd:element name="bill_address_id" type="xsd:integer"/>
<xsd:element name="ship_address_id" type="xsd:integer"/>
<xsd:element name="order_status" type="xsd:string"/>
<xsd:element name="credit_card_transaction" sql:is-constant="1">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="credit_card_type"/>
      <xsd:element ref="credit_card_number"/>
      <xsd:element ref="name_on_credit_card"/>
      <xsd:element ref="expiration_date"/>
      <xsd:element ref="authorization_id"/>
      <xsd:element ref="transaction_amount"/>
      <xsd:element ref="authorization_date"/>
      <xsd:element ref="transaction_country_id"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="credit_card_type" type="xsd:string"/>
<xsd:element name="credit_card_number" type="xsd:long"/>
<xsd:element name="name_on_credit_card" type="xsd:string"/>
<xsd:element name="expiration_date" type="xsd:dateTime"/>
<xsd:element name="authorization_id" type="xsd:string"/>
<xsd:element name="transaction_amount" type="xsd:decimal"/>
<xsd:element name="authorization_date" type="xsd:dateTime"/>
<xsd:element name="transaction_country_id" type="xsd:byte"/>
```

```
  <xsd:element name="order_lines" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="order_line"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="item_id" type="xsd:integer"/>
  <xsd:element name="quantity_of_item" type="xsd:integer"/>
  <xsd:element name="discount_rate" type="xsd:decimal"/>
  <xsd:element name="special_instructions" type="xsd:string"/>
  <xsd:element name="order_line" sql:relation="order_line_tab"
    sql:relationship="order_details" sql:key-fields="order_id order_line_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="item_id"/>
        <xsd:element ref="quantity_of_item"/>
        <xsd:element ref="discount_rate"/>
        <xsd:element ref="special_instructions"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="order_line_id"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

D.6 Annotated XSD for Customer

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:element name="YTD_payment" type="xsd:decimal"/>
  <xsd:element name="address_id" type="xsd:integer"/>
  <xsd:element name="balance" type="xsd:decimal"/>
  <xsd:element name="birth_date" type="xsd:dateTime"/>
  <xsd:element name="current_session_expiry" type="xsd:string"/>
  <xsd:element name="customer" sql:relation="customer_tab"
    sql:key-fields="customer_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="user_name"/>
        <xsd:element ref="password"/>
        <xsd:element ref="first_name"/>
        <xsd:element ref="last_name"/>
        <xsd:element ref="address_id"/>
        <xsd:element ref="phone_number"/>
        <xsd:element ref="email_address"/>
        <xsd:element ref="date_of_registration"/>
        <xsd:element ref="date_of_last_visit"/>
        <xsd:element ref="start_of_current_session"/>
        <xsd:element ref="current_session_expiry"/>
        <xsd:element ref="discount_rate"/>
        <xsd:element ref="balance"/>
        <xsd:element ref="YTD_payment"/>
        <xsd:element ref="birth_date"/>
        <xsd:element ref="miscellaneous_information"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="customer_id"/>
    </xsd:complexType>
```

```
  </xsd:element>
  <xsd:element name="customers" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="customer"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="date_of_last_visit" type="xsd:dateTime"/>
  <xsd:element name="date_of_registration" type="xsd:dateTime"/>
  <xsd:element name="discount_rate" type="xsd:decimal"/>
  <xsd:element name="email_address" type="xsd:string"/>
  <xsd:element name="first_name" type="xsd:string"/>
  <xsd:element name="last_name" type="xsd:string"/>
  <xsd:element name="miscellaneous_information" type="xsd:string"/>
  <xsd:element name="password" type="xsd:string"/>
  <xsd:element name="phone_number" type="xsd:long"/>
  <xsd:element name="start_of_current_session" type="xsd:string"/>
  <xsd:element name="user_name" type="xsd:string"/>
</xsd:schema>
```

D.7 Annotated XSD for Author

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSPY v5 U (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:element name="author" sql:relation="author_tab" sql:key-fields="author_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="first_name"/>
        <xsd:element ref="middle_name"/>
        <xsd:element ref="last_name"/>
        <xsd:element ref="date_of_birth"/>
        <xsd:element ref="biography"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="author_id"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="authors" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="author"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="biography" type="xsd:string"/>
  <xsd:element name="date_of_birth" type="xsd:dateTime"/>
  <xsd:element name="first_name" type="xsd:string"/>
  <xsd:element name="last_name" type="xsd:string"/>
  <xsd:element name="middle_name" type="xsd:string"/>
</xsd:schema>
```

D.8 Annotated XSD for Country

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:element name="countries" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="country"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="country" sql:relation="country_tab"
    sql:key-fields="country_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="name"/>
        <xsd:element ref="exchange_rate"/>
        <xsd:element ref="currency"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="country_id"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="currency" type="xsd:string"/>
  <xsd:element name="exchange_rate" type="xsd:decimal"/>
  <xsd:element name="name" type="xsd:string"/>
</xsd:schema>
```

D.9 Annotated XSD for Address

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sql="urn:schemas-microsoft-com:mapping-schema">
  <xsd:annotation>
    <xsd:appinfo>
      <sql:relationship name="street_details" parent="address_tab"
        parent-key="address_id" child="street_address_tab"
        child-key="address_id"/>
    </xsd:appinfo>
  </xsd:annotation>

  <xsd:element name="addresses" sql:is-constant="1">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="address" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="address" sql:relation="address_tab"
    sql:key-fields="address_id">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="street_address"/>
        <xsd:element ref="name_of_city"/>
        <xsd:element ref="name_of_state"/>
        <xsd:element ref="zip_code"/>
        <xsd:element ref="country_id"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:integer" sql:field="address_id"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="country_id" type="xsd:integer"/>
```

```
  <xsd:element name="name_of_city" type="xsd:string"/>
  <xsd:element name="name_of_state" type="xsd:string"/>
  <xsd:element name="street_address" sql:relation="street_address_tab"
    sql:relationship="street_details" sql:key-fields="address_id street_address"
    type="xsd:string"/>
  <xsd:element name="zip_code" type="xsd:string"/>
</xsd:schema>
```

Appendix E

SQL Server DDLs

E.1 SQL Server DDL for TCSD Class

```
CREATE TABLE [dbo].[dictionary_tab] (
    [entry_id] [nvarchar] (10) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[hw_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [hw] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[pr_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [pr] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[pos_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [pos] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[vd_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [vd] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[vf_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [vf] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO


CREATE TABLE [dbo].[et_cr_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [cr] [nvarchar] (1000) NOT NULL
```

```
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[s_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[def_cr_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL ,
    [cr] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[q_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL ,
    [q_id] [nvarchar] (10) NOT NULL ,
    [qd] [nvarchar] (1000) NULL ,
    [a] [nvarchar] (1000) NULL ,
    [w] [nvarchar] (1000) NULL ,
    [bib] [nvarchar] (1000) NULL ,
    [loc] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[qt_cr_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL ,
    [q_id] [nvarchar] (10) NOT NULL ,
    [cr] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[qt_i_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL ,
    [q_id] [nvarchar] (10) NOT NULL ,
    [i] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[qt_b_tab] (
    [entry_id] [nvarchar] (10) NOT NULL ,
    [s_id] [nvarchar] (10) NOT NULL ,
    [q_id] [nvarchar] (10) NOT NULL ,
    [b] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO


ALTER TABLE [dbo].[dictionary_tab] ADD
    CONSTRAINT [dictionary_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id]
    )  ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[hw_tab] ADD
    CONSTRAINT [hw_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [hw]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[pr_tab] ADD
    CONSTRAINT [pr_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [pr]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[pos_tab] ADD
    CONSTRAINT [pos_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [pos]
    )  ON [PRIMARY]
GO


ALTER TABLE [dbo].[vd_tab] ADD
    CONSTRAINT [vd_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [vd]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[vf_tab] ADD
    CONSTRAINT [vf_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [vf]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[et_cr_tab] ADD
    CONSTRAINT [et_cr_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [cr]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[s_tab] ADD
    CONSTRAINT [s_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[def_cr_tab] ADD
```

```
    CONSTRAINT [def_cr_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id],
        [cr]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[q_tab] ADD
    CONSTRAINT [q_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id],
        [q_id]
    )  ON [PRIMARY]
GO


ALTER TABLE [dbo].[qt_cr_tab] ADD
    CONSTRAINT [qt_cr_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id],
        [q_id],
        [cr]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[qt_i_tab] ADD
    CONSTRAINT [qt_i_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id],
        [q_id],
        [i]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[qt_b_tab] ADD
    CONSTRAINT [qt_b_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [entry_id],
        [s_id],
        [q_id],
        [b]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[hw_tab] ADD
    CONSTRAINT [hw_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[pr_tab] ADD
    CONSTRAINT [pr_tab_fk_1] FOREIGN KEY
```

```
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[pos_tab] ADD
    CONSTRAINT [pos_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO


ALTER TABLE [dbo].[vd_tab] ADD
    CONSTRAINT [vd_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[vf_tab] ADD
    CONSTRAINT [vf_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[et_cr_tab] ADD
    CONSTRAINT [et_cr_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[s_tab] ADD
    CONSTRAINT [s_tab_fk_1] FOREIGN KEY
    (
        [entry_id]
    ) REFERENCES [dbo].[dictionary_tab] (
        [entry_id]
    )
GO

ALTER TABLE [dbo].[def_cr_tab] ADD
    CONSTRAINT [def_cr_tab_fk_1] FOREIGN KEY
    (
        [entry_id],
        [s_id]
    ) REFERENCES [dbo].[s_tab] (
```

```
        [entry_id],
        [s_id]
    )
GO


ALTER TABLE [dbo].[q_tab] ADD
    CONSTRAINT [q_tab_fk_1] FOREIGN KEY
    (
        [entry_id],
        [s_id]
    ) REFERENCES [dbo].[s_tab] (
        [entry_id],
        [s_id]
    )
GO

ALTER TABLE [dbo].[qt_cr_tab] ADD
    CONSTRAINT [qt_cr_tab_fk_1] FOREIGN KEY
    (
        [entry_id],
        [s_id],
        [q_id]
    ) REFERENCES [dbo].[q_tab] (
        [entry_id],
        [s_id],
        [q_id]
    )
GO

ALTER TABLE [dbo].[qt_i_tab] ADD
    CONSTRAINT [qt_i_tab_fk_1] FOREIGN KEY
    (
        [entry_id],
        [s_id],
        [q_id]
    ) REFERENCES [dbo].[q_tab] (
        [entry_id],
        [s_id],
        [q_id]
    )
GO

ALTER TABLE [dbo].[qt_b_tab] ADD
    CONSTRAINT [qt_b_tab_fk_1] FOREIGN KEY
    (
        [entry_id],
        [s_id],
        [q_id]
    ) REFERENCES [dbo].[q_tab] (
        [entry_id],
        [s_id],
        [q_id]
    )
GO
```

E.2 SQL Server DDL for TCMD Class

```
CREATE TABLE [dbo].[article_tab] (
    [article_id] [int] NOT NULL ,
    [lang] [nvarchar] (1000) NULL ,
    [title] [nvarchar] (1000) NULL ,
    [city] [nvarchar] (1000) NULL ,
    [country] [nvarchar] (1000) NULL ,
    [date] [datetime] NULL ,
    [genre] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[article_author_tab] (
    [article_id] [int] NOT NULL ,
    [name] [nvarchar] (1000) NOT NULL ,
    [email] [nvarchar] (1000) NULL ,
    [phone] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[keyword_tab] (
    [article_id] [int] NOT NULL ,
    [keyword] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[body_p_tab] (
    [article_id] [int] NOT NULL ,
    [p] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ack_tab] (
    [article_id] [int] NOT NULL ,
    [pa] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[ref_tab] (
    [article_id] [int] NOT NULL ,
    [a_id] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[body_sec_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[sec_p_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [p] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[sec_subsec_tab] (
```

```
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[subsec_p_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL ,
    [p] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[subsec_subsubsec_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsec_heading] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[subsubsec_p_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsec_heading] [nvarchar] (1000) NOT NULL ,
    [p] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[subsubsec_subsubsubsec_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsubsec_heading] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[subsubsubsec_p_tab] (
    [article_id] [int] NOT NULL ,
    [sec_heading] [nvarchar] (1000) NOT NULL ,
    [subsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsec_heading] [nvarchar] (1000) NOT NULL ,
    [subsubsubsec_heading] [nvarchar] (1000) NOT NULL ,
    [p] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[article_tab] ADD
    CONSTRAINT [article_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[article_author_tab] ADD
    CONSTRAINT [article_author_tab_pk_1] PRIMARY KEY  CLUSTERED
```

```
    (
        [article_id],
        [name]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[keyword_tab] ADD
    CONSTRAINT [keyword_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [keyword]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[body_p_tab] ADD
    CONSTRAINT [body_p_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [p]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[ack_tab] ADD
    CONSTRAINT [ack_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [pa]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[ref_tab] ADD
    CONSTRAINT [ref_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [a_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[body_sec_tab] ADD
    CONSTRAINT [body_sec_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[sec_p_tab] ADD
    CONSTRAINT [sec_p_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [p]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[sec_subsec_tab] ADD
    CONSTRAINT [sec_subsec_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
```

```
        [sec_heading],
        [subsec_heading]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[subsec_p_tab] ADD
    CONSTRAINT [subsec_p_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [p]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[subsec_subsubsec_tab] ADD
    CONSTRAINT [subsec_subsubsec_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[subsubsec_p_tab] ADD
    CONSTRAINT [subsubsec_p_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading],
        [p]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[subsubsec_subsubsubsec_tab] ADD
    CONSTRAINT [subsubsec_subsubsubsec_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading],
        [subsubsubsec_heading]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[subsubsubsec_p_tab] ADD
    CONSTRAINT [subsubsubsec_p_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading],
        [subsubsubsec_heading],
        [p]
    )  ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[article_author_tab] ADD
    CONSTRAINT [article_author_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO


ALTER TABLE [dbo].[keyword_tab] ADD
    CONSTRAINT [keyword_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO

ALTER TABLE [dbo].[body_p_tab] ADD
    CONSTRAINT [body_p_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO

ALTER TABLE [dbo].[ack_tab] ADD
    CONSTRAINT [ack_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO

ALTER TABLE [dbo].[ref_tab] ADD
    CONSTRAINT [ref_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO

ALTER TABLE [dbo].[body_sec_tab] ADD
    CONSTRAINT [body_sec_tab_fk_1] FOREIGN KEY
    (
        [article_id]
    ) REFERENCES [dbo].[article_tab] (
        [article_id]
    )
GO

ALTER TABLE [dbo].[sec_p_tab] ADD
    CONSTRAINT [sec_p_tab_fk_1] FOREIGN KEY
    (
```

```
        [article_id],
        [sec_heading]
    ) REFERENCES [dbo].[body_sec_tab] (
        [article_id],
        [sec_heading]
    )
GO

ALTER TABLE [dbo].[sec_subsec_tab] ADD
    CONSTRAINT [sec_subsec_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading]
    ) REFERENCES [dbo].[body_sec_tab] (
        [article_id],
        [sec_heading]
    )
GO

ALTER TABLE [dbo].[subsec_p_tab] ADD
    CONSTRAINT [subsec_p_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading],
        [subsec_heading]
    ) REFERENCES [dbo].[sec_subsec_tab] (
        [article_id],
        [sec_heading],
        [subsec_heading]
    )
GO

ALTER TABLE [dbo].[subsec_subsubsec_tab] ADD
    CONSTRAINT [subsec_subsubsec_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading],
        [subsec_heading]
    ) REFERENCES [dbo].[sec_subsec_tab] (
        [article_id],
        [sec_heading],
        [subsec_heading]
    )
GO

ALTER TABLE [dbo].[subsubsec_p_tab] ADD
    CONSTRAINT [subsubsec_p_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading]
    ) REFERENCES [dbo].[subsec_subsubsec_tab] (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading]
    )
GO
```

```
ALTER TABLE [dbo].[subsubsec_subsubsubsec_tab] ADD
    CONSTRAINT [subsubsec_subsubsubsec_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading]
    ) REFERENCES [dbo].[subsec_subsubsec_tab] (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading]
    )
GO

ALTER TABLE [dbo].[subsubsubsec_p_tab] ADD
    CONSTRAINT [subsubsubsec_p_tab_fk_1] FOREIGN KEY
    (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading],
        [subsubsubsec_heading]
    ) REFERENCES [dbo].[subsubsec_subsubsubsec_tab] (
        [article_id],
        [sec_heading],
        [subsec_heading],
        [subsubsec_heading],
        [subsubsubsec_heading]
    )
GO
```

E.3 SQL Server DDL for DCSD Class

```
CREATE TABLE [dbo].[catalog_tab] (
    [item_id] [nvarchar] (10) NOT NULL ,
    [title] [nvarchar] (1000) NULL ,
    [date_of_release] [datetime] NULL ,
    [publisher_name] [nvarchar] (1000) NULL ,
    [publisher_name_of_city] [nvarchar] (1000) NULL ,
    [publisher_name_of_state] [nvarchar] (1000) NULL ,
    [publisher_zip_code] [nvarchar] (1000) NULL ,
    [publisher_country_name] [nvarchar] (1000) NULL ,
    [publisher_country_exchange_rate] [numeric](10, 6) NULL ,
    [publisher_country_currency] [nvarchar] (1000) NULL ,
    [publisher_FAX_number] [bigint] NULL ,
    [publisher_phone_number] [bigint] NULL ,
    [publisher_web_site] [nvarchar] (1000) NULL ,
    [subject] [nvarchar] (1000) NULL ,
    [description] [nvarchar] (1000) NULL ,
    [thumbnail_data] [nvarchar] (1000) NULL ,
    [image_data] [nvarchar] (1000) NULL ,
    [suggested_retail_price] [numeric](6, 2) NULL ,
    [suggested_retail_price_currency] [nvarchar] (1000) NULL ,
    [cost] [numeric](6, 2) NULL ,
    [cost_currency] [nvarchar] (1000) NULL ,
    [when_is_available] [datetime] NULL ,
```

```
    [quantity_in_stock] [tinyint] NULL ,
    [ISBN] [nvarchar] (1000) NULL ,
    [number_of_pages] [smallint] NULL ,
    [type_of_book] [nvarchar] (1000) NULL ,
    [length] [smallint] NULL ,
    [length_unit] [nvarchar] (1000) NULL ,
    [width] [smallint] NULL ,
    [width_unit] [nvarchar] (1000) NULL ,
    [height] [smallint] NULL ,
    [height_unit] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[catalog_author_tab] (
    [item_id] [nvarchar] (10) NOT NULL ,
    [first_name] [nvarchar] (1000) NOT NULL ,
    [middle_name] [nvarchar] (1000) NOT NULL ,
    [last_name] [nvarchar] (1000) NOT NULL ,
    [date_of_birth] [datetime] NULL ,
    [biography] [nvarchar] (1000) NULL ,
    [name_of_city] [nvarchar] (1000) NULL ,
    [name_of_state] [nvarchar] (1000) NULL ,
    [zip_code] [nvarchar] (1000) NULL ,
    [name_of_country] [nvarchar] (1000) NULL ,
    [phone_number] [bigint] NULL ,
    [email_address] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[catalog_author_address_tab] (
    [item_id] [nvarchar] (10) NOT NULL ,
    [first_name] [nvarchar] (1000) NOT NULL ,
    [middle_name] [nvarchar] (1000) NOT NULL ,
    [last_name] [nvarchar] (1000) NOT NULL ,
    [street_address] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[catalog_publisher_address_tab] (
    [item_id] [nvarchar] (10) NOT NULL ,
    [street_address] [nvarchar] (1000) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[catalog_related_item_tab] (
    [item_id] [nvarchar] (10) NOT NULL ,
    [related_item_id] [nvarchar] (10) NOT NULL ,
) ON [PRIMARY]
GO


ALTER TABLE [dbo].[catalog_tab] ADD
    CONSTRAINT [catalog_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[catalog_author_tab] ADD
```

```
    CONSTRAINT [catalog_author_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id],
        [first_name],
        [middle_name],
        [last_name]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[catalog_author_address_tab] ADD
    CONSTRAINT [catalog_author_address_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id],
        [first_name],
        [middle_name],
        [last_name],
        [street_address]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[catalog_publisher_address_tab] ADD
    CONSTRAINT [catalog_publisher_address_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id],
        [street_address]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[catalog_related_item_tab] ADD
    CONSTRAINT [catalog_related_item_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id],
        [related_item_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[catalog_author_tab] ADD
    CONSTRAINT [catalog_author_tab_fk_1] FOREIGN KEY
    (
        [item_id]
    ) REFERENCES [dbo].[catalog_tab] (
        [item_id]
    )
GO

ALTER TABLE [dbo].[catalog_publisher_address_tab] ADD
    CONSTRAINT [catalog_publisher_address_tab_fk_1] FOREIGN KEY
    (
        [item_id]
    ) REFERENCES [dbo].[catalog_tab] (
        [item_id]
    )
GO

ALTER TABLE [dbo].[catalog_related_item_tab] ADD
    CONSTRAINT [catalog_related_item_tab_fk_1] FOREIGN KEY
    (
        [item_id]
    ) REFERENCES [dbo].[catalog_tab] (
```

```
        [item_id]
    )
GO

ALTER TABLE [dbo].[catalog_author_address_tab] ADD
    CONSTRAINT [catalog_author_address_tab_fk_1] FOREIGN KEY
    (
        [item_id],
        [first_name],
        [middle_name],
        [last_name]
    ) REFERENCES [dbo].[catalog_author_tab] (
        [item_id],
        [first_name],
        [middle_name],
        [last_name]
    )
GO
```

E.4 SQL Server DDL for DCMD Class

```
CREATE TABLE [dbo].[address_tab] (
    [address_id] [int] NOT NULL ,
    [name_of_city] [nvarchar] (1000) NULL ,
    [name_of_state] [nvarchar] (1000) NULL ,
    [zip_code] [nvarchar] (1000) NULL ,
    [country_id] [int] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[author_tab] (
    [author_id] [int] NOT NULL ,
    [first_name] [nvarchar] (1000) NULL ,
    [middle_name] [nvarchar] (1000) NULL ,
    [last_name] [nvarchar] (1000) NULL ,
    [date_of_birth] [datetime] NULL ,
    [biography] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[country_tab] (
    [country_id] [int] NOT NULL ,
    [name] [nvarchar] (1000) NULL ,
    [exchange_rate] [numeric](10, 6) NULL ,
    [currency] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[customer_tab] (
    [customer_id] [int] NOT NULL ,
    [user_name] [nvarchar] (1000) NULL ,
    [password] [nvarchar] (1000) NULL ,
    [first_name] [nvarchar] (1000) NULL ,
    [last_name] [nvarchar] (1000) NULL ,
    [address_id] [int] NULL ,
    [phone_number] [bigint] NULL ,
    [email_address] [nvarchar] (1000) NULL ,
    [date_of_registration] [datetime] NULL ,
```

```
    [date_of_last_visit] [datetime] NULL ,
    [start_of_current_session] [nvarchar] (1000) NULL ,
    [current_session_expiry] [nvarchar] (1000) NULL ,
    [discount_rate] [numeric](3, 2) NULL ,
    [balance] [numeric](3, 2) NULL ,
    [YTD_payment] [numeric](5, 2) NULL ,
    [birth_date] [datetime] NULL ,
    [miscellaneous_information] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[item_tab] (
    [item_id] [int] NOT NULL ,
    [title] [nvarchar] (1000) NULL ,
    [author_id] [int] NULL ,
    [date_of_release] [datetime] NULL ,
    [name_of_publisher] [nvarchar] (1000) NULL ,
    [subject] [nvarchar] (1000) NULL ,
    [description] [nvarchar] (1000) NULL ,
    [thumbnail] [nvarchar] (1000) NULL ,
    [image] [nvarchar] (1000) NULL ,
    [suggested_retail_price] [numeric](6, 2) NULL ,
    [cost] [numeric](6, 2) NULL ,
    [when_is_available] [datetime] NULL ,
    [quantity_in_stock] [tinyint] NULL ,
    [ISBN] [nvarchar] (1000) NULL ,
    [number_of_pages] [smallint] NULL ,
    [type_of_book] [nvarchar] (1000) NULL ,
    [size_of_book] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[order_line_tab] (
    [order_id] [int] NOT NULL ,
    [order_line_id] [int] NOT NULL ,
    [item_id] [int] NULL ,
    [quantity_of_item] [int] NULL ,
    [discount_rate] [numeric](3, 2) NULL ,
    [special_instructions] [nvarchar] (1000) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[order_tab] (
    [order_id] [int] NOT NULL ,
    [customer_id] [int] NULL ,
    [order_date] [datetime] NULL ,
    [subtotal] [numeric](6, 2) NULL ,
    [tax] [numeric](6, 2) NULL ,
    [total] [numeric](6, 2) NULL ,
    [ship_type] [nvarchar] (1000) NULL ,
    [ship_date] [datetime] NULL ,
    [bill_address_id] [int] NULL ,
    [ship_address_id] [int] NULL ,
    [order_status] [nvarchar] (1000) NULL ,
    [credit_card_type] [nvarchar] (1000) NULL ,
    [credit_card_number] [bigint] NULL ,
    [name_on_credit_card] [nvarchar] (1000) NULL ,
    [expiration_date] [datetime] NULL ,
    [authorization_id] [nvarchar] (1000) NULL ,
```

```
    [transaction_amount] [numeric](6, 2) NULL ,
    [authorization_date] [datetime] NULL ,
    [transaction_country_id] [tinyint] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[related_item_tab] (
    [related_item_id] [int] NOT NULL ,
    [item_id] [int] NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[street_address_tab] (
    [street_address] [nvarchar] (1000) NOT NULL ,
    [address_id] [int] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[address_tab] ADD
    CONSTRAINT [address_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [address_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[author_tab] ADD
    CONSTRAINT [author_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [author_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[country_tab] ADD
    CONSTRAINT [country_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [country_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[customer_tab] ADD
    CONSTRAINT [customer_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [customer_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[item_tab] ADD
    CONSTRAINT [item_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[order_line_tab] ADD
    CONSTRAINT [order_line_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [order_id],
        [order_line_id]
    )  ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[order_tab] ADD
    CONSTRAINT [order_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [order_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[related_item_tab] ADD
    CONSTRAINT [related_item_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [item_id],
        [related_item_id]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[street_address_tab] ADD
    CONSTRAINT [street_address_tab_pk_1] PRIMARY KEY  CLUSTERED
    (
        [address_id],
        [street_address]
    )  ON [PRIMARY]
GO

ALTER TABLE [dbo].[order_line_tab] ADD
    CONSTRAINT [order_line_tab_fk_1] FOREIGN KEY
    (
        [order_id]
    ) REFERENCES [dbo].[order_tab] (
        [order_id]
    )
GO

ALTER TABLE [dbo].[related_item_tab] ADD
    CONSTRAINT [related_item_tab_fk_1] FOREIGN KEY
    (
        [item_id]
    ) REFERENCES [dbo].[item_tab] (
        [item_id]
    )
GO

ALTER TABLE [dbo].[street_address_tab] ADD
    CONSTRAINT [street_address_tab_fk_1] FOREIGN KEY
    (
        [address_id]
    ) REFERENCES [dbo].[address_tab] (
        [address_id]
    )
GO
```

Appendix F

Error Messages

We get the following error message when we run the original queries Q10 and Q11 for 1GB indexed DC/SD database.

```
Q10:

for $a in input()/catalog/:item
where $a/date_of_release gt
"1990-01-01" and
    $a/date_of_release lt "1995-01-01"
return
    <Output>
        {$a/title}
        {$a/publisher}
    </Output>
sort by (publisher/name)

Q11:

for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
    $a/date_of_release lt "1995-01-01"
return
    <Output>
        {$a/title}
        {$a/date_of_release}
    </Output>
sort by ((date_of_realse) descending)

Q14:

for $a in input()/catalog/:item
where $a/date_of_release gt "1990-01-01" and
           $a/date_of_release lt "1991-01-01" and
           empty($a/publisher/contact_information/FAX_number)
return
      <Output>
        {$a/publisher/name}
      </Output>

Error Message:

java.lang.ClassCastException: xhive.XHIVE0518
    at xhive.XHIVE0961.compare(XHIVE0961:86)
```

```
at com.objy.pm.ooScalableCollectionsPersistor.cpp_compare
    (ooScalableCollectionsPersistor.java:353)
at com.objy.pm.ExternalInterface.jooBTree_containsWithCompare
    (Native Method)
at com.objy.pm.ExternalInterface.ooScalableCollectionsPersistor_
    contains(ExternalInterface.java:2075)
at com.objy.pm.ooSetPersistor.contains(ooSetPersistor.java:77)
at com.objy.db.util.ooBTree.contains(ooBTree.java:134)
at xhive.XHIVE0261.a(XHIVE0261:77)
at xhive.XHIVE0397.ba(XHIVE0397:1982)
at xhive.XHIVE0397.y(XHIVE0397:2015)
at xhive.XHIVE0397.y(XHIVE0397:2012)
at xhive.XHIVE0397.L(XHIVE0397:2024)
at xhive.XHIVE0397.A(XHIVE0397:2063)
at xhive.XHIVE0397.u(XHIVE0397:2105)
at xhive.XHIVE0397.bt(XHIVE0397:2130)
at xhive.XHIVE0397.s(XHIVE0397:2211)
at xhive.XHIVE0397.a(XHIVE0397:579)
at xhive.XHIVE0987.e(XHIVE0987:18)
at xhive.XHIVE0694.c(XHIVE0694:21)
at xhive.XHIVE1020.a(XHIVE1020:35)
at xhive.XHIVE0987.e(XHIVE0987:18)
at xhive.XHIVE0694.c(XHIVE0694:21)
at xhive.XHIVE0115.a(XHIVE0115:143)
at xhive.XHIVE0987.e(XHIVE0987:18)
at xhive.XHIVE0694.c(XHIVE0694:21)
at xhive.XHIVE0295.a(XHIVE0295:21)
at xhive.XHIVE0693.a(XHIVE0693:20)
at xhive.XHIVE0969.executeXQuery(XHIVE0969:487)
at xhive.XHIVE0969.executeXQuery(XHIVE0969:481)
at xhive.XHIVE0888.e(XHIVE0888:161)
at xhive.XHIVE0999.xhiveConstruct(XHIVE0999:299)
at com.xhive.adminclient.XhiveSwingWorker.construct
    (XhiveSwingWorker:51)
at com.xhive.adminclient.XhiveTransactedSwingWorker.construct
    (XhiveTransactedSwingWorker:57)
at com.xhive.adminclient.dialogs.SwingWorker$2.run(SwingWorker$2:119)
at java.lang.Thread.run(Thread.java:536)
```