History
Examples Of XML Documents

```xml
<invoice>
  <number>421</number>
  <date>2008-05-24</date>
  <items>
    <item>
      <description>Wool Paddock Shet Ret Double Bound Yellow 4'0"</description>
      <quantity>1</quantity>
      <unitPrice>105.00</unitPrice>
    </item>
    <item>
      <description>Wool Race Roller and Breastplate Red Double</description>
      <quantity>1</quantity>
      <unitPrice>75.00</unitPrice>
    </item>
    <item>
      <description>Paddock Jacket Red Size Medium Inc Embroidery</description>
      <quantity>2</quantity>
      <unitPrice>67.50</unitPrice>
    </item>
  </items>
</invoice>
```
Why Sedna?

Before

- **Non native** strategies - mapping an XML data model onto relational or object-oriented model

After

- **Native XML** – XML files are fundamental unit of storage
Native XML Database

- Human readable
- Store XML document as XML
- Describes data
- Easy to understand
- Self Describing
- Documents

XML

Native XML Database
Native XML Databases

sedna

MarkLogic

Native XML Database System

BASE

existdb

ORACLE

BERKELEY DB
### XML Model vs. Relational Model

#### Computer Table

<table>
<thead>
<tr>
<th>Id</th>
<th>Speed (MHz)</th>
<th>RAM (MB)</th>
<th>HD (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>800</td>
<td>256</td>
<td>40</td>
</tr>
<tr>
<td>102</td>
<td>933</td>
<td>512</td>
<td>40</td>
</tr>
</tbody>
</table>

```xml
<Table>
  <Computer Id='101'>
    <Speed>800Mhz</Speed>
    <RAM>256MB</RAM>
    <HD>40GB</HD>
  </Computer>
  <Computer Id='102'>
    <Speed>933Mhz</Speed>
    <RAM>512MB</RAM>
    <HD>40GB</HD>
  </Computer>
</Table>
```
Challenges Sedna Addresses

- Improved schema-based clustering storage strategy
- Novel memory management technique
- Not having to evaluate the special join operation
Sedna Features

- Open Source
- Native XML Database
- Based on the XQuery language and the XQuery/XPath data model
- XUpdate language
- SQL connection from XQuery
- Full-text search indices
- Support for ACID transactions
- Support for fine-grained XML triggers
Sedna Features

- Built-in Database Connection Pooling manager.
- Persistent storage
- Incremental hot backup
- Supports Unicode
- Zero dependencies
- Supports Binary BLOBS as well as Java Object storage.
- Database security
Why Move From Relational to Sedna

- Flexibility
- Scalability and Interoperability
- Performance increase
- No capacity limits for data type
- Searches: Structural and Semantical
Sedna Architecture

Figure 1: SEDNA Architecture Overview
Sedna Architecture
Sedna Native XML Database
Client/Server Protocol

- Message-based protocol for communication with clients through the TCP/IP sockets.
- Message Structure – First 4 – Instruction; Next 4 – Length; ‘Length ’ Bytes -Body
- To begin a session – start up message
- 3 types of queries – query, update, bulk
- Termination initiated by the client
<library>
  <book>
    <title>Foundation on databases</title>
    <author>Abiteboul</author>
    <author>Hull</author>
    <author>Vianu</author>
  </book>
  ... 
  <book>
    <title>An Introduction to Database Systems</title>
    <author>Date</author>
    <issue>
      <publisher>Addison-Wesley</publisher>
      <year>2004</year>
    </issue>
  </book>
  <paper>
    <title>A Relational Model for Large Shared Data Banks</title>
    <author>Codd</author>
  </paper>
  ... 
  <paper>
    <title>The Complexity of Relational Query Languages</title>
    <author>Codd</author>
  </paper>
</library>
Data Structure – Node Descriptor

- **title**: 
  - node handle
  - prev-in-block
  - left-sibling
  - label
  - children
  - next-in-block
  - right-sibling

- **parent**: 
  - Indirection table

- **prev**: 
  - in-block

- **in**: 
  - block

- **next**: 
  - in-block

- **right**: 
  - sibling

- **left**: 
  - sibling

- **nodes**: 
  - handle
  - label
  - children
Memory Management

- Database pointers need conversion from DAS (Database Address Space) to VAS (Virtual Address Space)
- Pointer swizzling creates and overhead cost
- SEDNA eliminates the need of swizzling
Memory Management

DATABASE ADDRESS SPACE: (layer, addr) addressing

ADDRESS RANGE (Ex: 0..4GB)

PROCESS VM

MAPPING REGION

BUFFERS

HDD

read, write

memory mapping
Sedna Memory Management

Client session

Database Address Space (DAS)

addr

Virtual Address Space (VAS)

MapViewOfFile (Windows)
mmap (Linux)

Buffer Manager

Buffer Memory

VirtualLock (Windows)
mlock (Linux)

Secondary Memory (Disk)
Advantages of Sedna Memory Management

- 64-bit virtual address space
- Pointer dereferencing is comparable to conventional pointers
- Avoids costly pointer swizzling
## Supported API Packages

<table>
<thead>
<tr>
<th>Language</th>
<th>Name</th>
<th>Type</th>
<th>Support</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>Sedna XQJ API</td>
<td>Driver</td>
<td>+</td>
<td>Charles Foster</td>
</tr>
<tr>
<td>Java</td>
<td>Sedna XML:DB API</td>
<td>Driver</td>
<td>+</td>
<td>Charles Foster</td>
</tr>
<tr>
<td>Python</td>
<td>Python API</td>
<td>Driver</td>
<td>+</td>
<td>Sedna Team</td>
</tr>
<tr>
<td></td>
<td>mod_sedna v.1.5</td>
<td>Apache HTTP Server Module</td>
<td>+</td>
<td>Sedna Team</td>
</tr>
<tr>
<td>Java</td>
<td>Sedna Admin</td>
<td>Admin GUI, Web Admin GUI</td>
<td>+</td>
<td>Flávio R. C. Sousa</td>
</tr>
<tr>
<td>Python</td>
<td>zif.sedna</td>
<td>Driver</td>
<td>+</td>
<td>Jim Washington</td>
</tr>
<tr>
<td>C#</td>
<td>.NET API (sources/svn)</td>
<td>Driver</td>
<td></td>
<td>Mindtouch, John Wood</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP API</td>
<td>Driver</td>
<td>+</td>
<td>Sedna Team</td>
</tr>
<tr>
<td>Scheme</td>
<td>Sedna API for Chicken Scheme</td>
<td>Chicken Egg</td>
<td>+</td>
<td>Felix Winkelmann</td>
</tr>
<tr>
<td>Python</td>
<td>SednaPy (DBAPI 2.0)</td>
<td>Driver</td>
<td>+</td>
<td>Steve Howe</td>
</tr>
<tr>
<td></td>
<td>SDBAdmin (for Windows)</td>
<td>Admin GUI</td>
<td>Tony Scott</td>
<td></td>
</tr>
<tr>
<td>Delphi</td>
<td>Delphi API for Sedna</td>
<td>Driver</td>
<td>+</td>
<td>Alexander Kardaisky</td>
</tr>
<tr>
<td>Ruby</td>
<td>Ruby API for Sedna</td>
<td>Driver</td>
<td>+</td>
<td>Rolf Timmermans</td>
</tr>
<tr>
<td>Perl</td>
<td>Perl API for Sedna</td>
<td>Driver</td>
<td>+</td>
<td>Daniel Ruoso</td>
</tr>
<tr>
<td>Catalyst</td>
<td>Catalyst Model for Sedna</td>
<td>Adapter</td>
<td>+</td>
<td>Daniel Ruoso</td>
</tr>
<tr>
<td>Component Pascal</td>
<td>BlackBox Subsystem for Sedna</td>
<td>Driver</td>
<td>+</td>
<td>Eugene Temirzalezaev</td>
</tr>
<tr>
<td>Haskell</td>
<td>Haskell Bindings</td>
<td>Driver</td>
<td>+</td>
<td>Eric Jones</td>
</tr>
</tbody>
</table>
JAVA API

Session

```java
SednaConnection getConnection(String url,
String DBName,
String user,
String password)
throws DriverException
```

Transaction

```java
public void begin() throws DriverException
public void commit() throws DriverException
public void rollback() throws DriverException
```

Statement

```java
public SednaStatement createStatement()
throws DriverException
public void loadDocument(InputStream in,
String doc_name)
throws DriverException, IOException
public void loadDocument(InputStream in,
String doc_name,
String col_name)
throws DriverException, IOException
public boolean execute(String queryText)
throws DriverException
```

Result

```java
public SednaSerializedResult getSerializedResult()
public String next() throws DriverException
```
DB GUI Interface

Support for Sedna 3.x.x
Validation XML (DTD, XSD)
Manage all aspects of databases
Show metadata (DataGuide)
Execute XQuery and update extensions
Platform Independent (Java)
Easy to install and configure

Features

SednaAdmin Interface
XQuery is a standard expression language modeled after and with syntax similar to SQL. It incorporates XPath Expression and FLWOR Expression. Not only for XML files, XQuery is designed to work with various data formats.
XQuery Functions

- Built-in functions
  - Xpath/XQuery function library, e.g., document()
  - Aggregation functions, e.g., avg, sum, count, max, min
- User-defined functions
  - Ex.: xq. Functions.xq

```xml
declare function prefix:functio_name($parameter as datatype?...) as returnDatatype?
{
  function body...
};
```
XQuery FLWOR Expression

for var in expr

let var := expr

where expr

order by expr

return expr
XPath?

Language using path expressions

FLWOR Expression

Addresses select parts of documents

Sequence of steps

Similar to Operating System file path schema
Operating Systems | XPath
--- | ---
/ = the root directory | /library = the root element (if named library)
/users/dave/foo = the (one) file named foo in dave in users | /library/book/chapter/section = every section element in a chapter in every book in the library
foo = the (one) file named foo in the current directory | section = every section element that is a child of the current element
. = the current directory | . = the current element
.. = the parent directory | .. = parent of the current element
/users/dave/* = all the files in /users/dave | /library/book/chapter/* = all the elements in /library/book/chapter
XQuery Triggers

- Specific to XML data
- Native based
- XML data hierarchy dependent
- Similar purposes as relational database triggers: integrity constraints, event-based applications, statistics gathering, monitoring specific data changes...
XQuery Triggers

CREATE TRIGGER "trigger-name"
(BEFORE | AFTER) (INSERT | DELETE | REPLACE)
ON <XPath-expression> (,<XPath-expression>)*
( FOR EACH NODE | FOR EACH STATEMENT )
DO
{
    (<XUpdate-expression($NEW, $OLD, $WHERE)>;)*
    <XQuery-expression($NEW, $OLD, $WHERE)>
}

DROP TRIGGER "trigger_name"
CREATE TRIGGER "tr1"
BEFORE INSERT
ON doc("auction")/site//person
FOR EACH NODE
DO
{
    if($NEW/age < 14)
        then
            <person>{attribute id {$NEW/@id}}
                {$NEW/*}
            <age-group>young</age-group>
    </person>
    else
        <person>{attribute id {$NEW/@id}}
            {$NEW/*}
        <age-group>adult</age-group>
    </person>
}
Transactions & Recovery

- ACID Transactions
  - Atomicity – rollback procedure
  - Consistency – by design
  - Isolation – S2PL and snapshots
  - Durability - two level recovery scheme

- Multiversioning – Concurrency Control
- Logging and Recovery
Multi-versioning Scheme

- **Page-Level Versioning**
  - Snapshot-based schema with data elements as pages.
  - Transaction transparent solution
  - All the logic is encapsulated in the storage manager.
  - No worry of garbage collection.

- **Read-Only Transactions**
  - Support faster execution for Read Only transactions (Queries)
  - Isolation not needed: Non-Blocking processing or non-S2PL
Logging & Recovery Mechanisms

- **Normal Processing**
  - Logging of all main operations such as insert node, create index, etc.
  - Transaction-consistent snapshot at checkpoints – *Persistent snapshot*

- **Rollback Processing**
  - Undo the operations using the data from the already created log.

- **Two-Level Recovery Process**
  - Restoring transaction-consistent state of DB using the persistent snapshot.
  - Redo the necessary operations using the log generated
User-Based Authorization System

- DBA user and ordinary user
- DB creator and DBA grant & revoke privileges
- Role: named group of related privileges
- Password Client Authentication

Image description: A diagram illustrating a user-based authorization system with interconnected roles and privileges.
Privileges

- CREATE USER
- CREATE DOCUMENT
- CREATE COLLECTION
- CREATE INDEX
- CREATE FT INDEX
- CREATE TRIGGER
- LOAD
- LOAD MODULE
- DROP
- RETRIEVE-METADATA
- INSERT
- QUERY
- DELETE
Sedna and its competitors

Sedna vs. X-Hive

- 100 MB XMark Benchmark
- AMD Athlon 64 2.00 GHz, 1 GB of RAM.
- Timeout: 2000

<table>
<thead>
<tr>
<th>Operation</th>
<th>X-Hive</th>
<th>Sedna</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td>1.2</td>
<td>0.8</td>
</tr>
<tr>
<td>XPath, pos, trans</td>
<td>4.0</td>
<td>1.7</td>
</tr>
<tr>
<td>Complex XPath</td>
<td>6.8</td>
<td>2.2</td>
</tr>
<tr>
<td>Id comparison</td>
<td>3.7</td>
<td>2.3</td>
</tr>
<tr>
<td>XPath, count</td>
<td>3.0</td>
<td>0.4</td>
</tr>
<tr>
<td>FLWR</td>
<td>4.6</td>
<td>0.5</td>
</tr>
<tr>
<td>FLWR, count</td>
<td>16.1</td>
<td>0.8</td>
</tr>
<tr>
<td>Join(1,2)</td>
<td>*</td>
<td>1046</td>
</tr>
<tr>
<td>Join(1,2,3)</td>
<td>*</td>
<td>1350</td>
</tr>
<tr>
<td>Group by</td>
<td>34.8</td>
<td>81</td>
</tr>
<tr>
<td>Semijoin</td>
<td>*</td>
<td>1664</td>
</tr>
<tr>
<td>Complex semijoin</td>
<td>*</td>
<td>373</td>
</tr>
<tr>
<td>Struct. XPath + trans</td>
<td>3.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Contains substring</td>
<td>10.4</td>
<td>8.4</td>
</tr>
<tr>
<td>Long XPath</td>
<td>1.8</td>
<td>0.1</td>
</tr>
<tr>
<td>Nested Long XPath</td>
<td>2.3</td>
<td>0.7</td>
</tr>
<tr>
<td>Empty</td>
<td>3.1</td>
<td>2.1</td>
</tr>
<tr>
<td>Function Calls</td>
<td>2.6</td>
<td>1.0</td>
</tr>
<tr>
<td>Sorting</td>
<td>24.3</td>
<td>3.5</td>
</tr>
<tr>
<td>Trans(nested XPaths)</td>
<td>3.3</td>
<td>2.5</td>
</tr>
</tbody>
</table>
## Sedna and its competitors

### Sedna vs. Berkeley XML DB

- 12MB XMark benchmark
- AMD Athlon 64 2.00 GHz, 1 GB of RAM.
- Timeout: 2000

<table>
<thead>
<tr>
<th>Operation</th>
<th>BDB node</th>
<th>Sedna</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td>0.172</td>
<td>0.109</td>
</tr>
<tr>
<td>XPath, pos, trans</td>
<td>0.421</td>
<td>0.188</td>
</tr>
<tr>
<td>Complex XPath</td>
<td>0.625</td>
<td>0.141</td>
</tr>
<tr>
<td>Id comparison</td>
<td>0.969</td>
<td>0.250</td>
</tr>
<tr>
<td>XPath, count</td>
<td>0.188</td>
<td>0.094</td>
</tr>
<tr>
<td>FLWR</td>
<td>1.297</td>
<td>0.109</td>
</tr>
<tr>
<td>FLWR, count</td>
<td>7.016</td>
<td>0.172</td>
</tr>
<tr>
<td>Join(1,2)</td>
<td>263.219</td>
<td>11.109</td>
</tr>
<tr>
<td>Join(1,2,3)</td>
<td>428.453</td>
<td>14.125</td>
</tr>
<tr>
<td>Group by</td>
<td>42.250</td>
<td>2.219</td>
</tr>
<tr>
<td>Semijoin</td>
<td>281.781</td>
<td>34.625</td>
</tr>
<tr>
<td>Complex semijoin</td>
<td>81.453</td>
<td>10.969</td>
</tr>
<tr>
<td>Struct. XPath, trans</td>
<td>0.109</td>
<td>0.456</td>
</tr>
<tr>
<td>Contains substring</td>
<td>3.797</td>
<td>2.485</td>
</tr>
<tr>
<td>Long XPath</td>
<td>0.219</td>
<td>0.047</td>
</tr>
<tr>
<td>Nested Long XPath</td>
<td>0.234</td>
<td>0.156</td>
</tr>
<tr>
<td>Empty</td>
<td>0.312</td>
<td>0.125</td>
</tr>
<tr>
<td>Function Calls</td>
<td>*</td>
<td>0.062</td>
</tr>
<tr>
<td>Sorting</td>
<td>*</td>
<td>0.43</td>
</tr>
<tr>
<td>Trans(nested XPaths)</td>
<td>1.016</td>
<td>0.156</td>
</tr>
</tbody>
</table>
XML Databases - Comparison

DB-Engines Ranking of Native XML DBMS

© November 2016, DB-Engines.com
Where To Use Sedna?
How big is it today?

User Reviews

**Tagivan**

🌟🌟🌟🌟

We've been using Sedna XML DB on Production for about a year as a part of a big product management system based on the xml data model. We're quite happy with this solution although we'd still like to see some more features available (like importing XQuery modules from the local file system resource or from a jar).

Posted 12/01/2011

**externalreality**

🌟🌟🌟🌟

Few realize how powerful and flexible an XML database actually is and Sedna is a strong offering in the category. Its free, fast, and flexible. Many man hours went into this XML and it shows. Good thought went into the document store and it certainly shows aswell.

Posted 09/15/2011

**mozinsur**

🌟🌟🌟🌟

Very cool Native-XML Database with triggers.

Posted 08/31/2009
Why Sedna Failed to Compete?

<table>
<thead>
<tr>
<th>Rank</th>
<th>DBMS</th>
<th>Database Model</th>
<th>Score Nov 2016</th>
<th>Score Oct 2016</th>
<th>Score Nov 2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>MarkLogic</td>
<td>Multi-model</td>
<td>10.22</td>
<td>-0.08</td>
<td>-0.78</td>
</tr>
<tr>
<td>2.</td>
<td>Virtuoso</td>
<td>Multi-model</td>
<td>2.56</td>
<td>-0.13</td>
<td>-1.00</td>
</tr>
<tr>
<td>3.</td>
<td>Sedna</td>
<td>Native XML DBMS</td>
<td>1.34</td>
<td>+0.12</td>
<td>-0.07</td>
</tr>
<tr>
<td>4.</td>
<td>Tamino</td>
<td>Native XML DBMS</td>
<td>1.05</td>
<td>+0.02</td>
<td>-0.07</td>
</tr>
<tr>
<td>5.</td>
<td>BaseX</td>
<td>Native XML DBMS</td>
<td>0.97</td>
<td>+0.05</td>
<td>-0.20</td>
</tr>
<tr>
<td>6.</td>
<td>eXist-db</td>
<td>Native XML DBMS</td>
<td>0.20</td>
<td>-0.00</td>
<td>-0.05</td>
</tr>
<tr>
<td>7.</td>
<td>searchxml</td>
<td>Multi-model</td>
<td>0.00</td>
<td>-0.01</td>
<td></td>
</tr>
</tbody>
</table>
Why Sedna Failed to Compete?

- General weakness of XML
- Switching from RDBMS is expensive
- Frozen further development
- Lack of support of Xquery 3.0
- Doesn’t support XML 1.1
- Scalability is supported but decreases the performance
## Competitors

<table>
<thead>
<tr>
<th>Name</th>
<th>XQJ</th>
<th>XML:DB</th>
<th>RESTful</th>
<th>RESTXQ</th>
<th>WebDAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseX</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>eXist</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MarkLogic Server</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Qizx</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Sedna</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
References

http://elib.mi.sanu.ac.rs/files/journals/kjm/30/kjom3013.pdf
https://www.sedna.org