Extensible databases as middleware for flexible information management

Karl Aberer
GMD-IPSI, Dolivostr. 15, 64239 Darmstadt
aberer@darmstadt.gmd.de
www.darmstadt.gmd.de/IPSI

DAGSTUHL Seminar on „System Integration“
15.3. - 19.3.99
Overview

Introduction
• ORDBMS

Integration Aspects
• Integrated processing of (single) data types
• Integration of (different) data types
• Integration of information sources

Summary
• Reasons to use ORDBMS for integration purposes

_disclaimer_
• _subjective view derived from actual work_
• _no transactional aspects considered here_
Object-Relational Database Systems

- user-defined data types
- user-defined functions
- index and optimizer extensions
- packaging: DataBlades, Cartridges, Extenders

```
<table>
<thead>
<tr>
<th>VRML</th>
<th>XML</th>
<th>DataBlade API</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPEG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIO DB</td>
<td>CLF DB</td>
<td></td>
</tr>
<tr>
<td>WEB DB</td>
<td>TXT DB</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DAS Data Manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Extensible SQL3 Parser</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ExtensibleOptimizer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParallelFunctionManager</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParallelAccessMethods</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ParallelDASDataManager</td>
</tr>
</tbody>
</table>
```
Extensible Databases - The Perspectives

universal storage of information
- store and integrate all possible data types
- integrate generic application logic with data
- integrate information from different sources

opposed to the universal access to information
DataBlade Examples

- Congress-online: a Web application
  - pure Web DataBlade application
- GPCRDB: information integration application
  - simple application specific extensions (layout, conversion)
- ICE: Web application development
  - specific data structures and functions
- MPEG: media metadata management
  - integration of complex parsing functions
- VRML: 3D visualisation
  - integration of complex parsing/manipulation functions
- XML document stores: combining structure and content
  - integration of complex parsing/manipulation functions and data structures
- CLF: presentation management
  - tightly coupled MMDBMS architecture
Integration Aspect 1: (DB centric)
integrate all functions in the life cycle of a data type

persistent/transient (BLOBS/streams/uninterpreted documents)

layouting/presenting

application
language

views

transient

SQL/functions

querying

database

persistent (structured data)

parsing/analysing/annotating

data sources

structural data metadata

transient

transient

SQL/functions

storing
Welcome to ECCMID Congress-Online!

If you are visiting ECCMID Congress-Online for the first time, click here for identification.
If you have visited ECCMID Congress-Online before, click here to log in with your already existing identification.

Standard IUD + Web-DataBlade Application
many users: > 3000, online access
abstract submission, registration, information
individual user profiles
ICE - Information Catalogue Environment

<?icemenu3 type="dynamic" width="1" dim0="$dim1" dim1="$dim1" dim2="$dim2" language="$language" page="$page">
VRML DataBlade

VRML dynamic node

SQL node
sql = select f1,f2,... from DB
where C

subgraph

f4 f2 f1 f3

VRML nodes

VRMLExpand()

EXECUTE: SELECT f1,f2,... FROM DB
WHERE C

Results

| Fields | f1  | f2  | f3  | f4  | ...
|--------|-----|-----|-----|-----|-----
| Row 1  | V_{1,1} | V_{1,2} | V_{1,3} | V_{1,4} | ... |
| Row 2  | V_{2,1} | V_{2,2} | V_{2,3} | V_{2,4} | ... |

...

generate VRML

Group node

result row

result row

2

V_{2,4} V_{2,2} V_{2,1} V_{2,3}

SQL extensions
- VRML datatype
- VRML manipulation and retrieval functions
- VRML dynamic nodes
- import/export of VRML data
Information life cycle - example VRML

- Persistent/transient (BLOBs/streams)
- VRML explode
- VRML scenes
  - SQL Nodes in VRML scenes
  - Combinations of VRML objects and data
    - Scene selection
    - Translation/rotation
  - Transient
  - Querying
  - Transient
  - SRML objects
    - STQL
    - Function API
    - Persistent (structured data)
    - Storing
OR Modelling Challenges

materialization of (expensive reading) functions

- update frequency and query access pattern
- provide exclusively functions
  - user decides which data he needs to materialize
  - has to take care of consistency
- provide data structures
  - materializes (often) large amounts of unnecessary data
  - inflexible
execute procedure MPEGCreateVideo()

MPEG Parser

VideoServer access

ParserState access

VideoStream
VideoSequence
SequenceHeader
GroupOfPicture
Picture
Slice
MacroBlock
Block

calls
reads
stores
XML document stores

Select all plays where Bernardo appears as a speaker in a scene.
XML document stores - Implementation issues

- index structures
  - document paths
  - element properties

- optimization of physical document representation

- Navigation on flat elements
  - complete markup (XML)
  - linear scans instead of recursive descend (get all elements of type)

---

The Tragedy of Hamlet, Prince of Denmark

<fm> ... </fm>
<personae>
<personae title> Dramatis personae </personae title>
<playsubt>
....
</playsubt>

<acttitle> Act I </acttitle>
<scene>
....
</scene>
Experimental Results

Documents from Hamlet play, one document ~ 300 KB

- retrieve all elements of a single document

<table>
<thead>
<tr>
<th>getAll (1 document)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
<th>fine fragm. (5000 x 0.22 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE (20-elements)</td>
<td>2,0 s</td>
<td>16,7 s</td>
<td>52,0 s</td>
<td>4609,0 s</td>
</tr>
<tr>
<td>STAGEDIR (134 elements)</td>
<td>1,0 s</td>
<td>16,3 s</td>
<td>53,3 s</td>
<td>4546,0 s</td>
</tr>
<tr>
<td>LINE (4014 elements)</td>
<td>1,7 s</td>
<td>17,0 s</td>
<td>53,3 s</td>
<td>4556,0 s</td>
</tr>
</tbody>
</table>

- retrieve all elements from all documents

<table>
<thead>
<tr>
<th>getObjs (9 documents)</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT (60 KB)</td>
<td>21,0 sec</td>
<td>3,5 sec</td>
<td>3,5 sec</td>
</tr>
<tr>
<td>SCENE (15 KB)</td>
<td>21,0 sec</td>
<td>71,0 sec</td>
<td>4,0 sec</td>
</tr>
</tbody>
</table>

- navigation at element level

<table>
<thead>
<tr>
<th>10 x succ(succ(succ(succ()))))</th>
<th>totally flat (1 x 300 KB)</th>
<th>rough fragm. (5 x 60 KB)</th>
<th>medium fragm. (22 x 15 KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCENE</td>
<td>12,0 sec</td>
<td>5,0 sec</td>
<td>15,0 sec</td>
</tr>
<tr>
<td>LINE</td>
<td>8,0 sec</td>
<td>4,0 sec</td>
<td>3,0 sec</td>
</tr>
</tbody>
</table>
Observations

If you have a function provide the function!
- encapsulation principle
- optimized data structures in addition or by user

Don’t interpret every materialized data structure!
- store data representation as BLOB
- provide it to the functions
- MPEG, VRML, XML

If you can’t index don’t query!
- efficient evaluation of not indexed functions in search predicates is a problem
- extensibility of query optimization?
Implementation Aspects of a Tightly Coupled Architecture

+ use of SQL, indexing, caching, multiuser
+ maintenance
  • single point of maintenance
+ robustness
  • industrial strength implementation
– overhead
  • design time
    – changes hurt, preferrably implement stable types -> standards
    – only useful for frequently used/DBMS system developers task (HTML, MPEG)
  • runtime because of data management function
– robustness/scalability
  • single point of failure/access
– vendor dependency
Integration Aspect 2: integrate different data types
Continuous Long Fields DataBlade

Application

SQL access

Continuous Long Fields DataBlade

Contin. Object Management client

CLF DataBlade

Contin. Transport Module client

Discrete data

Continuous data

CLF DataBlade

Contin. Transport Module server

External Storage Manager

CLF DataBlade

(DBMS part)

Video Foundation DataBlade

Informix Server
CLF schema
Interoperability between database extensions

- at the client: e.g. VRML and Web
- at SQL level: default
- at function call level: e.g. XML
  - indirect via SQL
  - direct
- at data structure: e.g. CLF Blade
Aspects of integration of different data types

semantics: different models for co-processing
- at client: server gives no support
- at SQL
  - join data
  - optimization possible
- at function level: common programming environment

function calls
- functions via SQL: type safety, automatic environment
- direct function calls: more efficient

DataBlade development
- if function call/refs to data structures are used no independent parallel development possible
- hierarchies of DataBlades
Integration Aspect 3: integrate heterogeneous autonomous data sources

Commonalities with previous cases
  • views vs. materialization
    • materialization exploits existing processing caps
    • views require query decomposition
    • OR „too heavy“ for virtual views
  • interoperability at SQL level

Differences
  • autonomy
  • integration from behaviorally „closed“ models, heterogeneity at data schema level
  • (in)stability of wrappers
Example: GPCRDB

SELECT fa* FROM sw, famtree fa WHERE TRANSMEM1 matches "*KA*" AND sw ac = fa ac INTO TEMP selected.

Summary of Query Results

Totally 7 matches found in the following families

<table>
<thead>
<tr>
<th>#</th>
<th>Family</th>
<th>Number of hits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class A Rhodopsin like</td>
<td>Prostanoid</td>
</tr>
<tr>
<td>2</td>
<td>Class A Rhodopsin like</td>
<td>Viral</td>
</tr>
<tr>
<td>3</td>
<td>Class A Rhodopsin like</td>
<td>Orphan/other</td>
</tr>
</tbody>
</table>

To drill down into a specific family, please click one of the family names listed above. You may also select to view ALL the results irrespective of their families.

Please try our Smart Query Engine that relaxes your query intelligently (and select the facets that you won't want relaxation - if any).

Data integration, efficient storage, querying, sequence specific functions, query relaxation, WWW access.
GPCRDB Architecture

**CLIENT**
- Web Browser
  - Query pages (HTML forms)
  - JavaScripts

**SERVER**
- Web Server
- IU S Web Driver
- SMART ENGINE
- File Databases
- Perl Scripts

Query request: Query results (in HTML pages)
Example: MIRO-WEB Architecture

- **Browser**
- **Java API**
- **Http Server**
- **Message Manager and Translator**
- **DISCO**
- **SCHEMA INTEGRATOR**
- **ORACLE 8**
- **Adapter**

Layers:
- **Client Layer**
- **Mediator Layer**
- **Materialized Layer**
- **Server Layer**

Connections:
1. From Browser to Java API
2a. From Java API to Http Server
2b. From Http Server to Java API
3. From Java API to Message Manager and Translator
4. From Message Manager and Translator to ORACLE 8
5. From SCHEMA INTEGRATOR to ORACLE 8
6. From DISCO to SCHEMA INTEGRATOR
7. From SCHEMA INTEGRATOR to ORACLE 8
8. From ORACLE 8 to SCHEMA INTEGRATOR
9. From SCHEMA INTEGRATOR to Message Manager and Translator
10. From Message Manager and Translator to DISCO
Non-Example: XML Broker

XML Browser

Query-Processor

XML Warehouse

JEDI Parser

Web Monitor

Query

Integrated XML

conditions

XML

XML

HTML

HTML

HTTP

Source DTD

 extraction grammar

Source description

Internetsources
Role of ORDBMS for data integration

wrappers as database extensions?
- implemented in an earlier Bio-project (Reliwe) -> too static
- not realized in GPCRDB, Miroweb

ORDBMS as mediator!
- if (mostly relational) data and query model appropriate
- if data is materialized

ORDBMS as presentation generator!
- business as usual

ORDBMS as repository!!!
Information life cycle

- **Heterogeneous databases (WWW)**
  - Wrapping
  - Layouting/presenting
  - HTML, VRML with embedded SQL
- **Relational data**
  - Function APIs
  - SQL
- **Integrated database**
  - Querying = mediating
  - Materializing
- **Persistent (structured data)**
  - Persistent/transient
  - Transient
  - Transient

- **Integrated views**
  - Transient
  - HTML, VRML with embedded SQL
  - SQL3

- **Persistent/transient**
  - Layouting/presenting
  - Wrapping
Which is better?
model semistructured data as OR or vice versa

if data is materialized
  • for traditional applications
    • object-relational with semistructured extensions
    • useful for co-existence
    • data warehousing as classical example
  • for the Web (mostly semi-structured data)
    • native database implementations for XML!
      (more flexibility, no fixed schemas required etc.)
    • yet missing: behavioral extensibility
  • thus
    • choose a canonical data model
    • take the best native implementation for storage

if data is not materialized
  • prefer light-weight implementations
Summary:
Non-Reasons to Use ORDBMS

1. use the DBMS as universal integration layer (do everything in the DBMS, universal storage)
   - DBMS requirements (e.g. schema) often inadequate
   - specialized solutions can be more efficient (e.g. XML)
   - only for relational data reasonable (e.g. Warehouses, but ...)

2. efficient development platform
   - complex
   - overhead
Possible Reasons

1. manage large amounts of **complexly structured** data efficiently
   - modelling flexibility without adequate efficient storage useless
2. process all kinds of complex queries efficiently
   - same as 1
here everything depends on
   - extensibility of query processing (new index structures, cost models, semantic optimization, etc.)
   - flexible strategies to deal with the parameterized functions
Possible Reasons

and an important non-technical reason

- relational is our standard platform ....
Definite Reasons

1. support presentation of operational data
   • success of Web, VRML etc. Blades
2. manage large amounts of derived/annotated relational data efficiently
   • use functions to derive (relational) data - storage is application specific
   • annotations to complex objects
Finally: ORDBMS is a moving target

different implementations
  • IBM DB2 ≠ IUS ≠ Oracle 8
lightweight implementations
  • Sybase, Oracle etc
hybrid implementations
  • Tamino (SAG)
Acknowledgements

Thomas Klement, Michael Fuchs, Yangjun Chen, Dunren Che, Weimin Chen, Hannelore Eisner, Silvia Hollfelder, Florian Schmidt, Thomas Risse, Matthias Malsy, Matthias Hemmje, Martin Leissler, Jürgen Wäsch, Andreas Stenger, Peter Fankhauser, Ralph Busse, Martin Mierse, ...