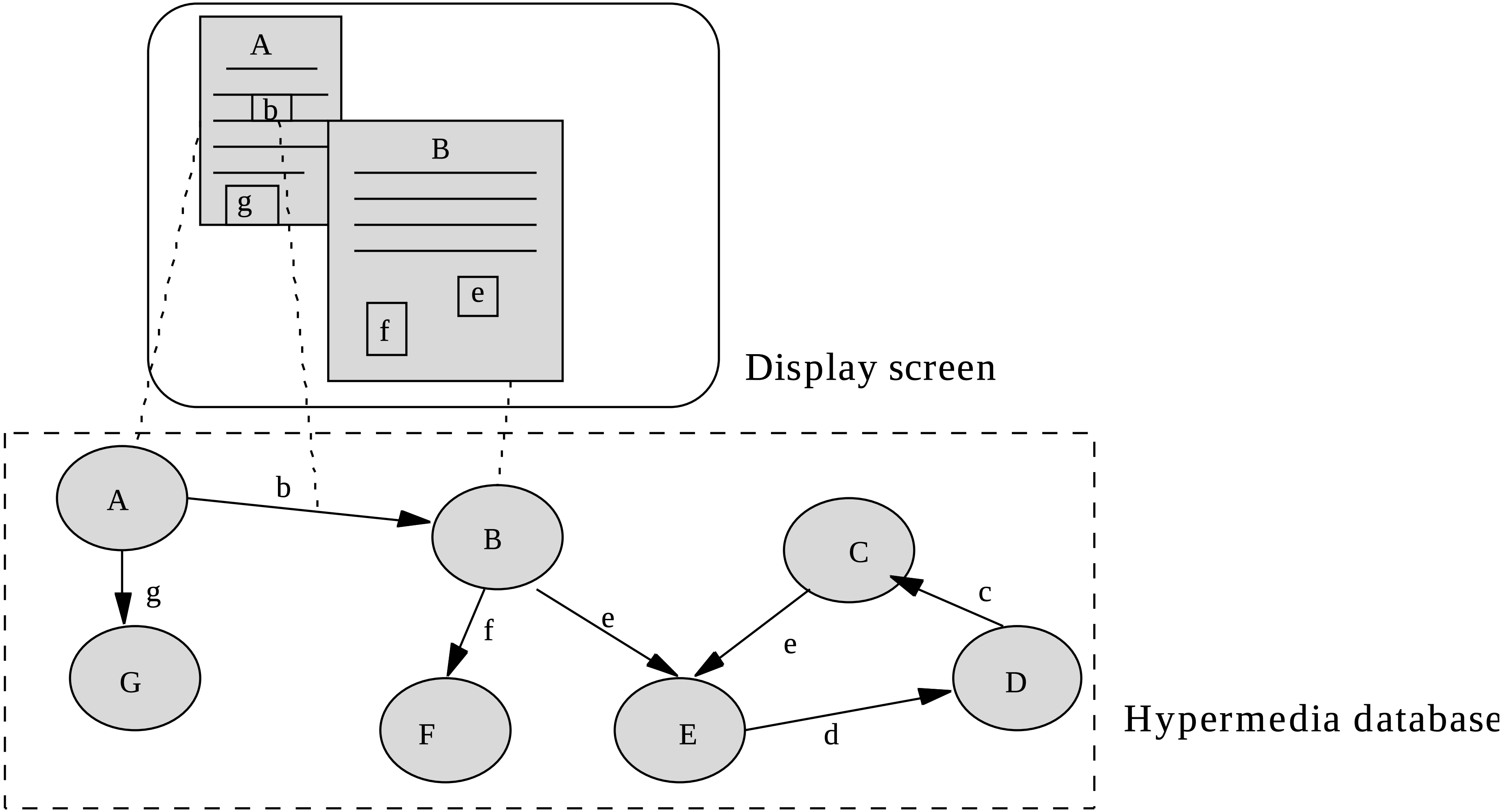
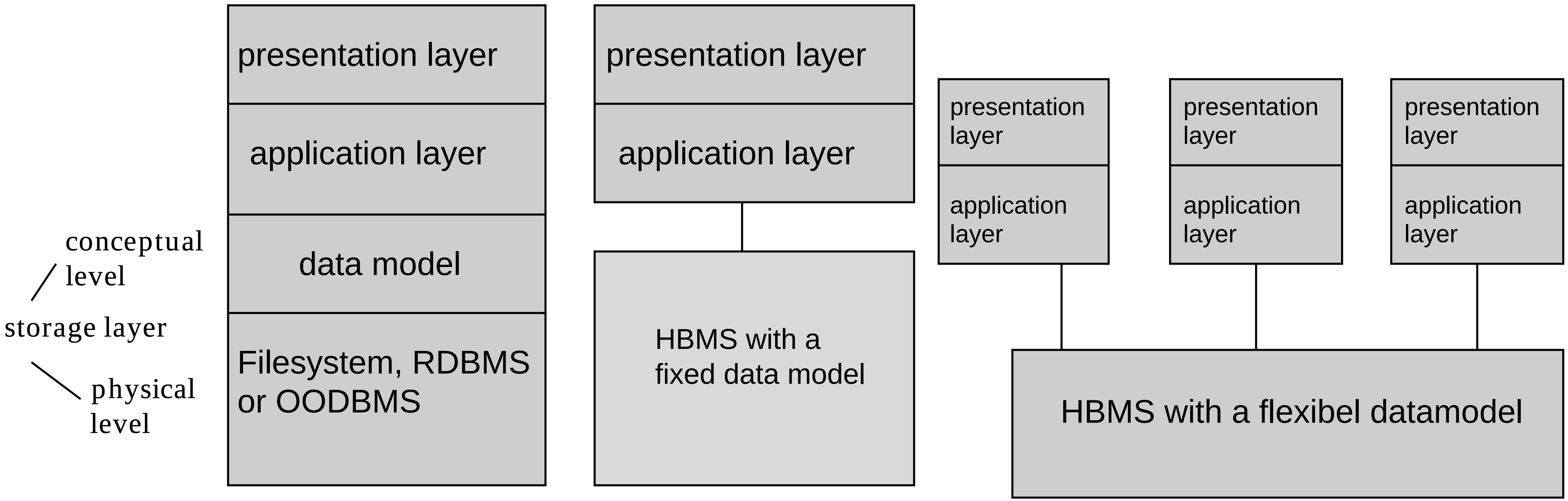
Hypermedia databases, Hypermedia storage

Hypermedia database



Hypertext system architectures



Current generation:
Hypertext system backend, monolithic (Intermedia, NoteCards, KMS...)

HAM generation:
HBMS with fixed data model, monolithic (HAM, HyperBase ....)

Next generation:
HBMS with flexibel data model, non-monolithic (Hyperform)

Overview

- Database requirements of hypermedia systems, OO concepts and hypermedia
- **The Intermedia hypertext system**: a current generation hypertext system, developed in 1986 at the Brown University
  - Concepts, architecture
  - Implementation on INGRES (RDBMS) and ENCORE (OODBMS)
  - Comparison of the two database system approaches in the context of hypermedia systems
- **HAM (Hypertext Abstract Machine)**: a general purpose hypertext storage system (HBMS), developed in 1988 at Tektronix' Computer Research Laboratory
  - Concepts, architecture
  - Application: implementation of NoteCards
- **Hyperform**: an open, extensible general purpose hypertext storage system (HBMS), developed in 1992 at the University of Aalborg and the Texas A&M University
  - Concepts, architecture
  - Applications

Characteristics of hypertext systems

- **Openness**
- **Collaborative work**
- **Data integrity**
- **Dynamism/Virtual structures**
- **Search and query mechanism**
- **Composites**
- **Versioning**
- **Multimedia**
- **Extensibility/Tailorability**

Hypermedia characteristics and the resulting requirements for the storage layer

- **Openness**

  The storage layer should provide

  – **uniformly stored data**: It should be possible to freely exchange data between different information systems.
- **Collaborative work/Sharing/Simultaneous multiuser access**

  The storage layer should provide

  – **consistency**: Long transactions and flexible locking protocols for concurrency control are needed.

  – **access control**: Object-specific access rights and additional access rights for annotation are needed.

  – **notification control**

  – **distribution** (in relation to performance and reliability)

Hypermedia characteristics and the resulting requirements for the storage layer

- **Data integrity/Correctness**

  The storage layer should provide

  - **traditional secondary storage management and data administration facilities.**
  - **constraint based integrity control** (for the contents of nodes and their structure)
  - **garbage collection:** This is more complicated. A non-referenced hypertext node is not garbage.
- **Dynamism/Virtual structures for dealing with changes**

  The storage layer has to implement

  - **virtual nodes, links and composites** (like views in RDBMS)

Hypermedia characteristics and the resulting requirements for the storage layer

- **<u>Search and query mechanism</u>**

  The storage layer should provide

  – **a query language**: Content search, property search and structure search is needed.

- **<u>Composites</u>**

  The data model should support

  – **composites.**

- **<u>Versioning</u>**

  The storage layer should support

  – **version control of information contents.** (requires a lot of space)

  – **version control of structure.** (is a “new” problem)

Hypermedia characteristics and the resulting requirements for the storage layer

- **<u>Multimedia</u>**

  The storage layer should be able to

  - **store and retrieve multimedia objects**: BLOBs are unacceptable. Specific access structures and compression algorithms are needed.
  - **access transparently different storage media**.
  - **query multimedia objects.**

- **<u>Extensibility and tailorability</u>**

  The storage layer should handle

  - **extensions** to the existing data model.

Evaluation of storage mechanisms

	Filesystem	RDBMS	OODBMS
Openness	**partly**	**yes**	**yes**
Sharing	**partly**	**partly**	**yes**
Integrity	**no**	**yes**	**yes**
Multimedia	**no**	**no**	**yes**
Querying	**partly**	**yes**	**partly**
Versioning	**partly**	**no**	**partly**
Extensibility	**no**	**yes**	**yes**

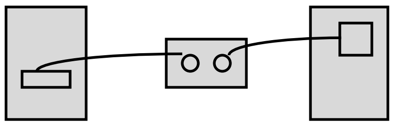
Results of Evaluation at University of Tokyo

OO concepts and Hypermedia

- Simple nodes can be compared to atomic objects
- Nodes can be accessed using node identifiers (object identifiers)
- A link can be represented by a set of at least 2 object identifiers.
- Links can be treated as objects with their own identifiers (link identifiers)
- A composite node can be treated as a composite object.
- ....
- ....

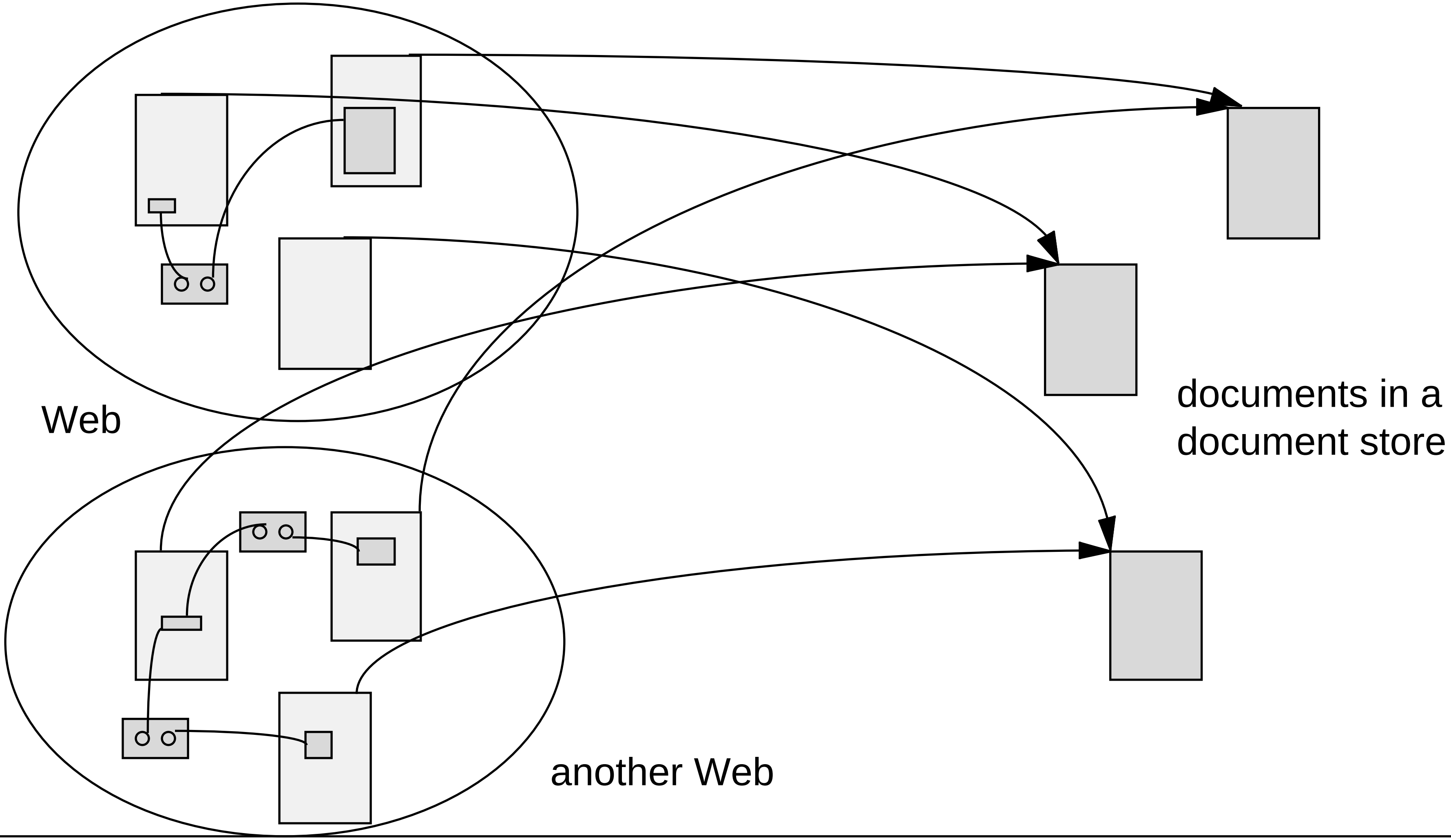
Intermedia: Concepts

- Bidirectional links connect two blocks (anchor point, specific location in a document)



- blocks and links have properties like f. ex. keywords, owner etc.
- Links are not global.
- Webs maintain the block and link information
- one web = one context

Intermedia: Webs



Intermedia: Concepts and implementation

Concepts:

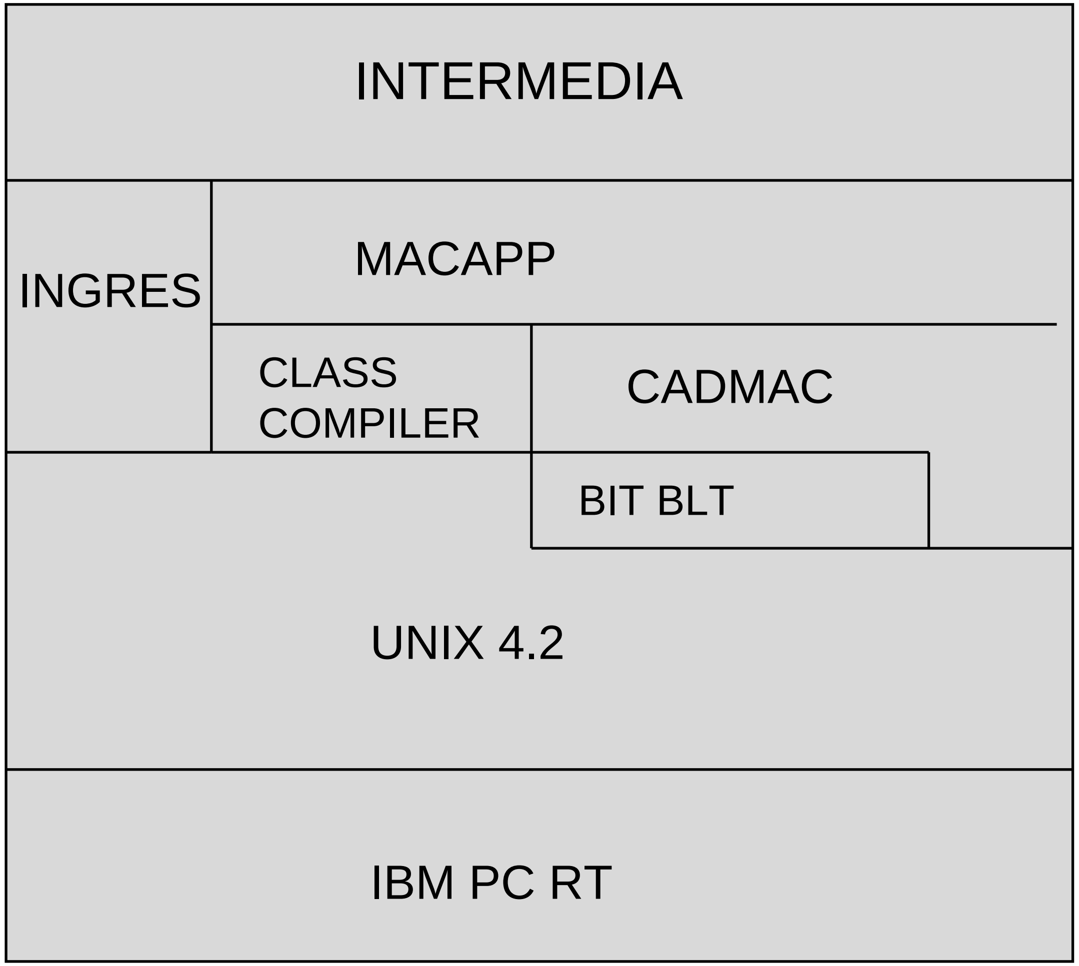
- Blocks
- Links
- Webs

Implementation:

- Block class
- Link class
- Web class

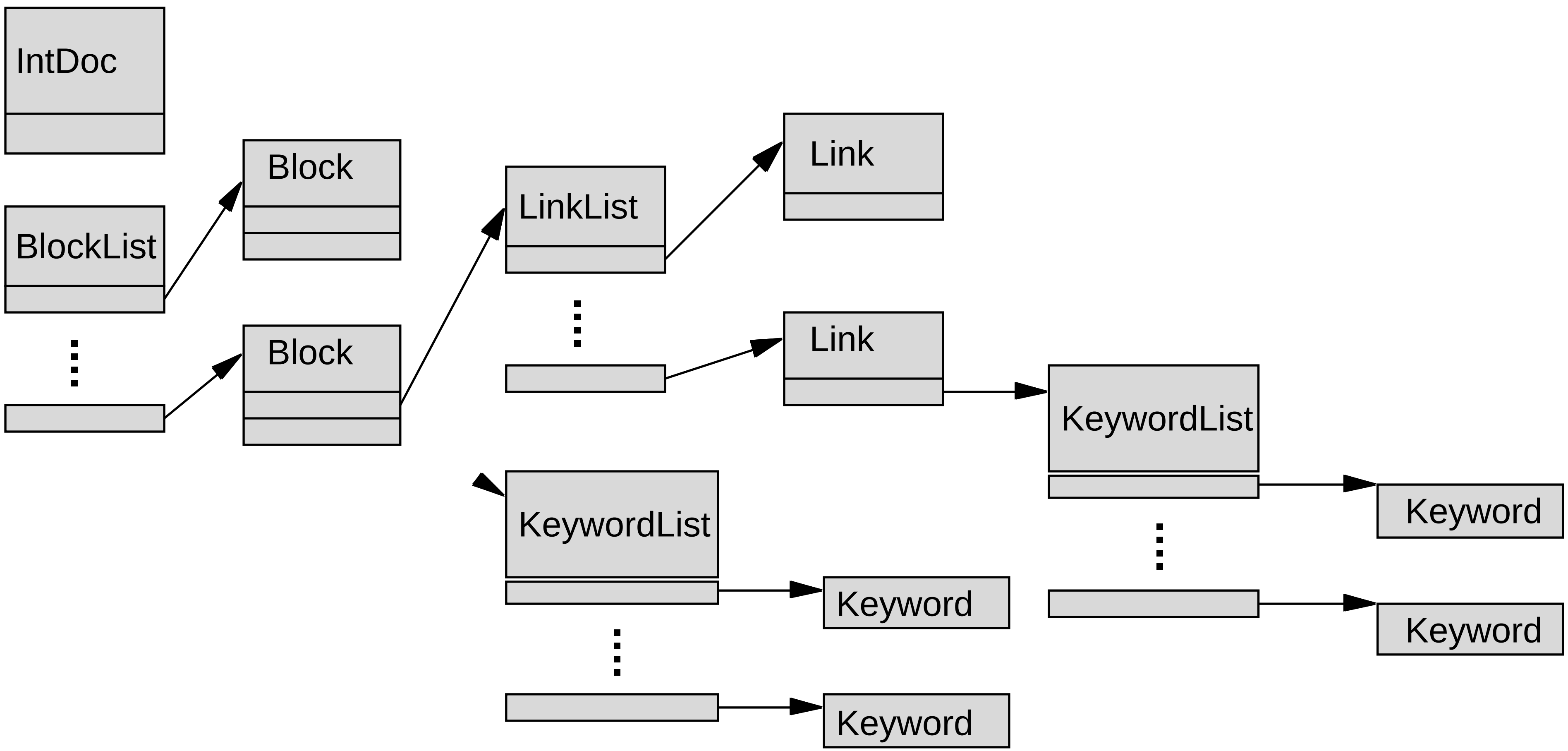
Intermedia is implemented in an object-oriented programming language called “Inheritance C”.

Intermedia: Architecture



The Intermedia layer

Intermedia: Object hierarchy



Intermedia: Link class definition

```
CLASS LINK (Object)

bool        fNEW;
bool        fChanged Keyword;
bool        fChanged Explain;
bool        fDeleted;
long        fCreateDate;
long        fModifyDate;
char        fAuthor;
short       fExplainExtend;         <-  flag used by INGRES
TExplain    fExplainer;
TList       fKeywordList;
short       fLinkId;                <- INGRES key
short       fType;
short       fSrcBlockId;            <- INGRES key
short       fSrcDocId;              <- INGRES key
short       fDestBlockId;           <- INGRES key
short       fDestDocId;             <- INGRES key
TBlock      fSrcBlockHand;
TBlock      fDestBlockHand;

PROCEDURES
            ILink();
            CompleteLink();
            UnLink();
TObject     Clone() OVERRIDE;
END
```

Intermedia: Database requirements

- A document may appear in more than one web.
- Multiple users (on different workstations) may access the same webs and documents simultaneously.
- Editing the contents of a document requires careful updating of block position information for all the webs that references the document.
- Data integrity must be maintained not only for the contents of the documents but also for their block and link information.

Intermedia: The database

solution:

- all the information is stored in a single system-wide database. This database should provide network-wide retrieval and update of information with appropriate concurrency control.
  - INGRES, an RDBMS is chosen
  - another RDBMS or an OODBMS could have been chosen
- The existence of the database is made transparent through block,link and web classes.

INGRES, an RDBMS

INGRES maintains a

- relation for each web
- relation for all the blocks of a web
- relation for all the links of a web

INGRES provides

- concurrency control
- locking mechanism

Intermedia:
INGRES relations that model the link class

Link Relation

Link Id	Type	DocID1	BlockID1	DocID2	BlockID2	Explainer	EExtend	CreateDate	ModifyDate	Owner
63	2	399	28	180	35	connecting a scanned reproduct	87	861008	870219	kes
64	1	399	28	352	40	explanation	0	861115	861115	kes

LExtend Relation

Link Id	Sequence	Explainer
63	1	ion of Titian’s “Venus and Ado
63	2	nis” to Titian’s biography

KeyWord Relation

KeyID	KeyWord
20	Titian
21	Venetian

KeyLink Relation

LinkID	KeyID
64	20
64	39

LinkFree Relation

Start	End
19	19
65	32767

KeyFree Relation

Start	End
120	32767

Intermedia: INGRES: Storing and retrieving objects

```
CLASS LINK (Object)

bool        fNEW;
bool        fChanged Keyword;
bool        fChanged Explain;
bool        fDeleted;
long        fCreateDate;
long        fModifyDate;
char        fAuthor;
short       fExplainExtend;
TExplain    fExplainer;
TList       fKeywordList;
short       fLinkId;
short       fType;
short       fSrcBlockId;
short       fSrcDocId;
short       fDestBlockId;
short       fDestDocId;
TBlock      fSrcBlockHand;
TBlock      fDestBlockHand;

PROCEDURES
            ILink();
            CompleteLink();
            UnLink();
TObject     Clone() OVERRIDE;
END
```

Storing →

The query language copies variables, that must be saved in a database record

← Retrieving

Allocating memory, The query language reconstruct the object

Link relation

LExtend Relation

KeyFree Relation

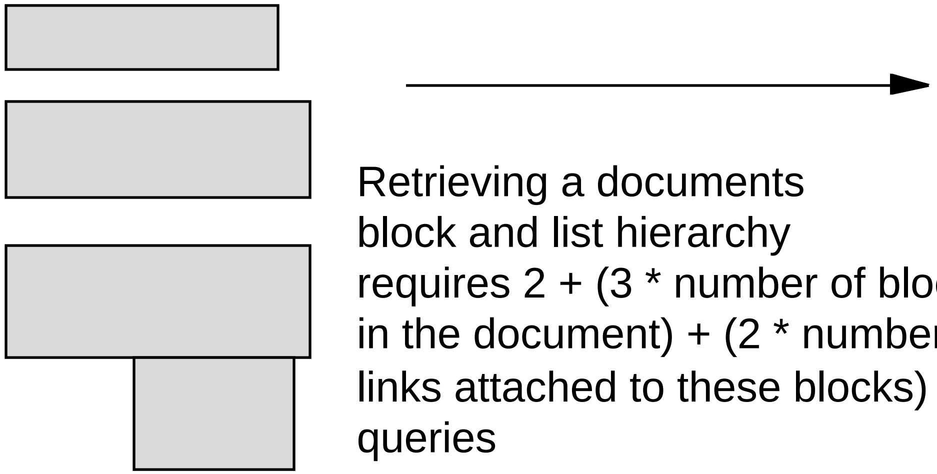
Keyword Relation

KeyLink Relation

LinkFree Relation

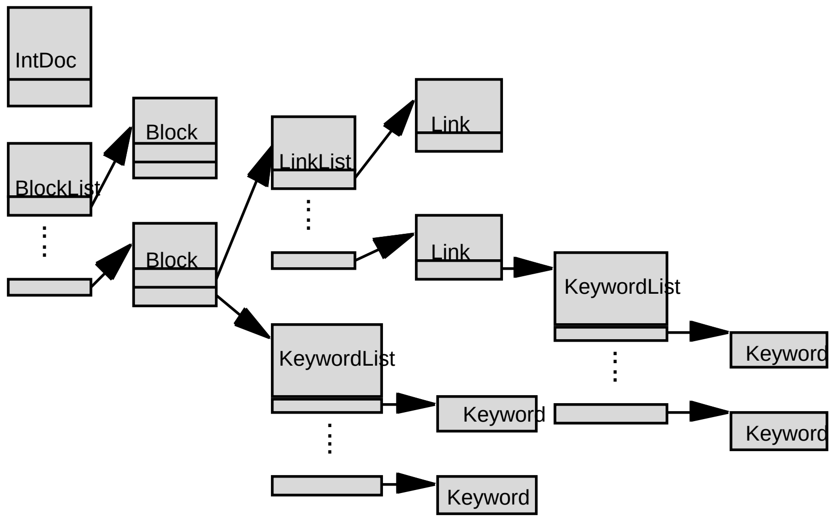
Intermedia:
INGRES: Retrieving a documents’ block and link hierarchy

INGRES relations



Retrieving a documents block and list hierarchy requires 2 + (3 * number of blocks in the document) + (2 * number of links attached to these blocks) queries

Intermedia Object hierarchy



ENCORE, an OODBMS

- Encore is based on a set of programmer defined classes.
- The objects are stored directly in the database. The application maintains only references to objects.
- Entire objects are written to the database.
- An object or a set of related objects (f.ex. an object hierarchy) can be locked.

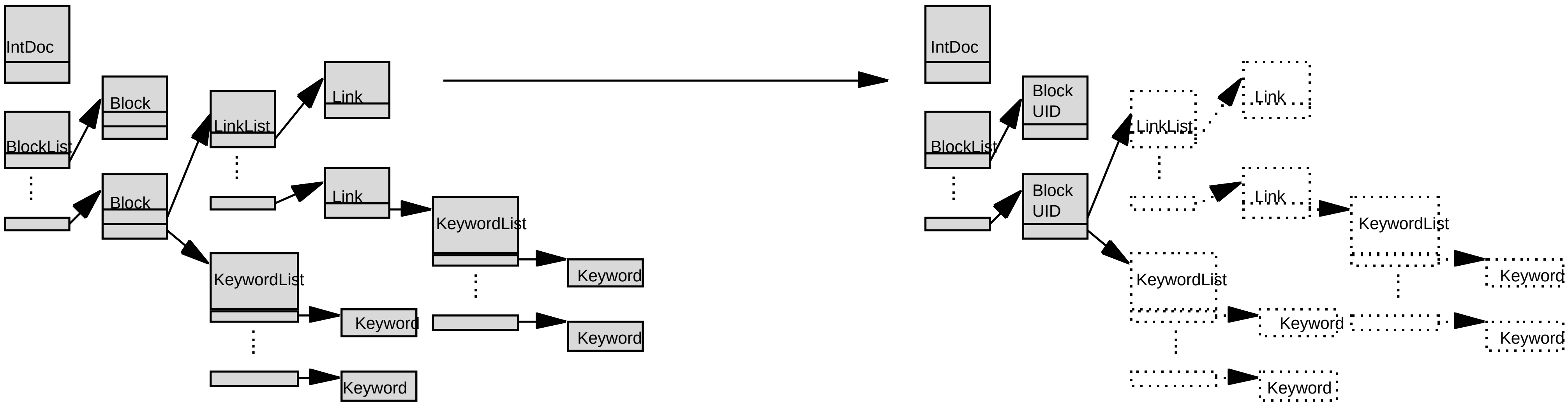
Define Class Link
Superclasses: Object
Properties:
block1: Block
block2: Block
explainer: Text
create_date: Date
modify_date: Date
owner: User_ID
keywords: set of Text
Methods:
initialize (L: Link)
ollow (L: Link, B: Block)

Intermedia:

ENCORE: Retrieving a documents’ block and list hierarchy

Encore

Intermedia Object hierarchy



Comparison: RDBMS and OODBMS

Relational DBMS

- Hierarchies to be flattened into relations
- High level of impedance mismatch between the prog. language data structures and the database data structures.
- Objects retrieved through query language.
- Many queries are needed to retrieve a single object.
- Queries are sequential.
- Results from queries to be stored in application's data structure.

Object-Oriented DBMS

- Hierarchies can be represented as they are
- Lower level of impedance mismatch
- Operations are performed on objects stored in the database.
- One message sent to an object can replace many queries.
- Objects send messages to other objects.
- Messages can directly manipulate data in the database and return a pointer to the result

Comparison: RDBMS and OODBMS

Relational DBMS

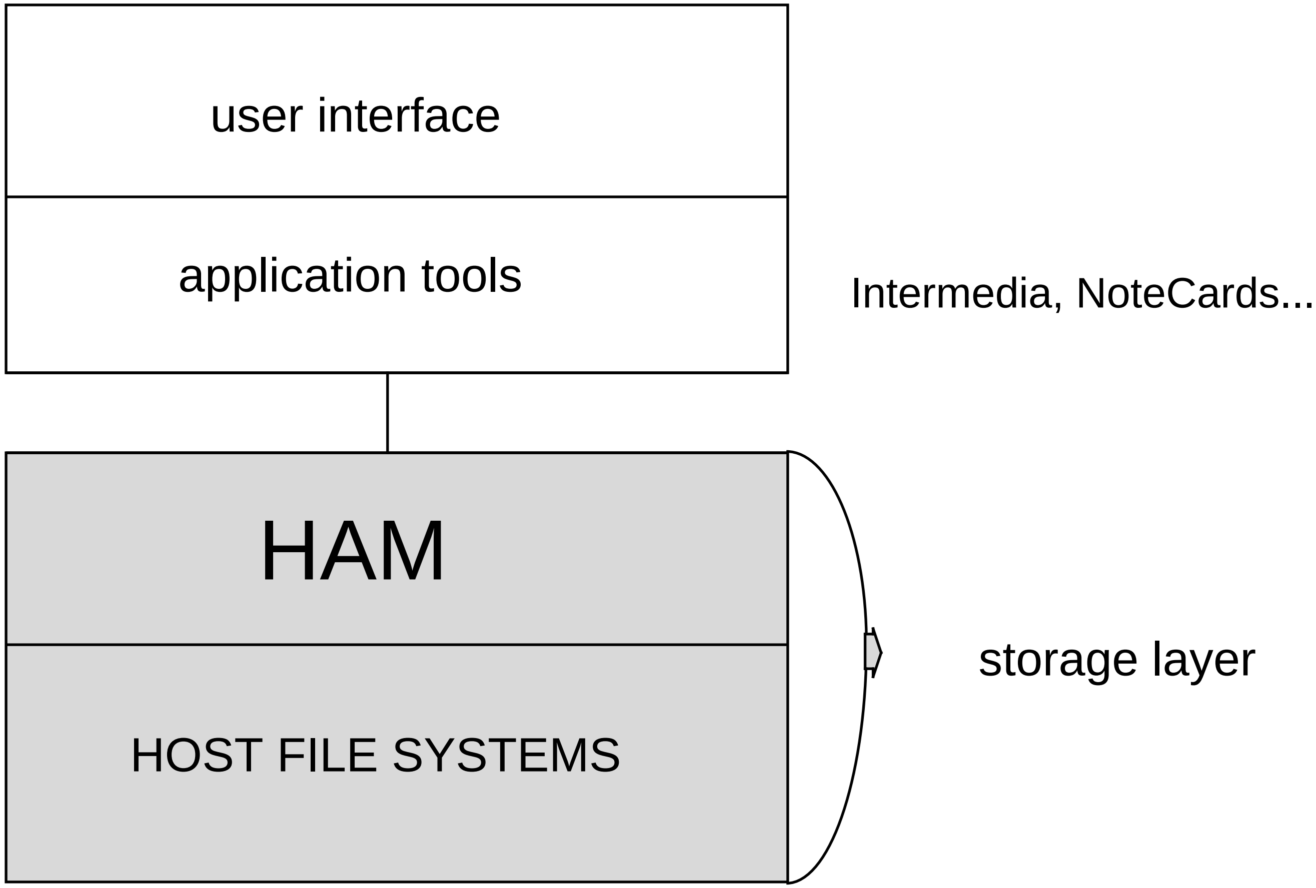
- Integrity checks enforced only during commit time
- Due to the flat nature of relations each record in each relation in a given hierarchy to be explicitly locked
- Bad performance even for simple link traversals (interactions with the database are expensive)
- Not very good support for collaborative work such as version management and concurrency control

Object-Oriented DBMS

- Integrity checks performed during run-time through triggers
- An entire hierarchy can be locked in a single operation
- Navigation model of the database can be directly exploited
- Flexible transaction processing facilities and wide range of locking policies
- OODBMS could also be used to store all the persistent data, like text and graphics objects.

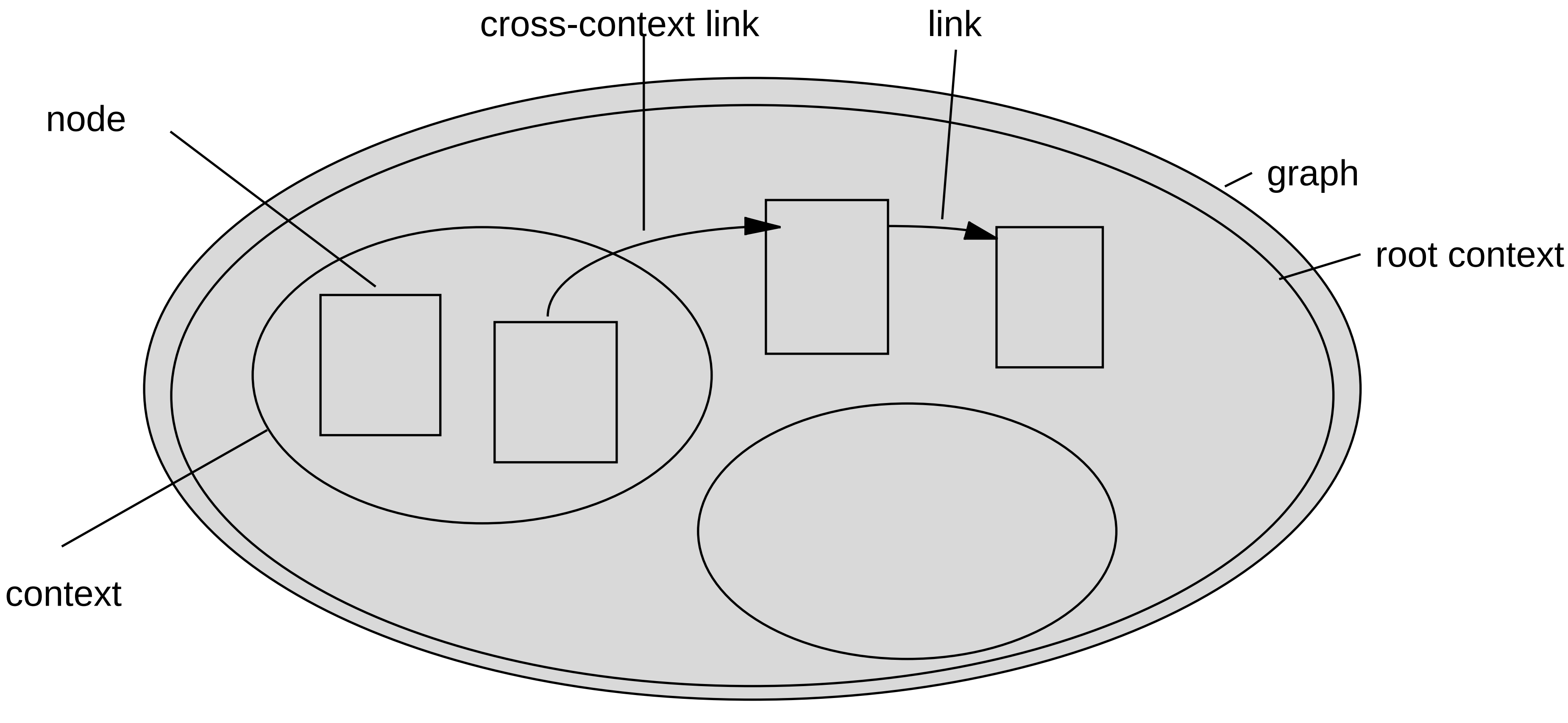
The Hypertext Abstract Machine (HAM)

a general purpose, transaction-based, multi-user server for a hypertext storage system



The HAM storage model: Objects

- **graphs**, **contexts**, **nodes**, **links**
- contexts, nodes and links have **attributes**



The HAM storage model: Operations

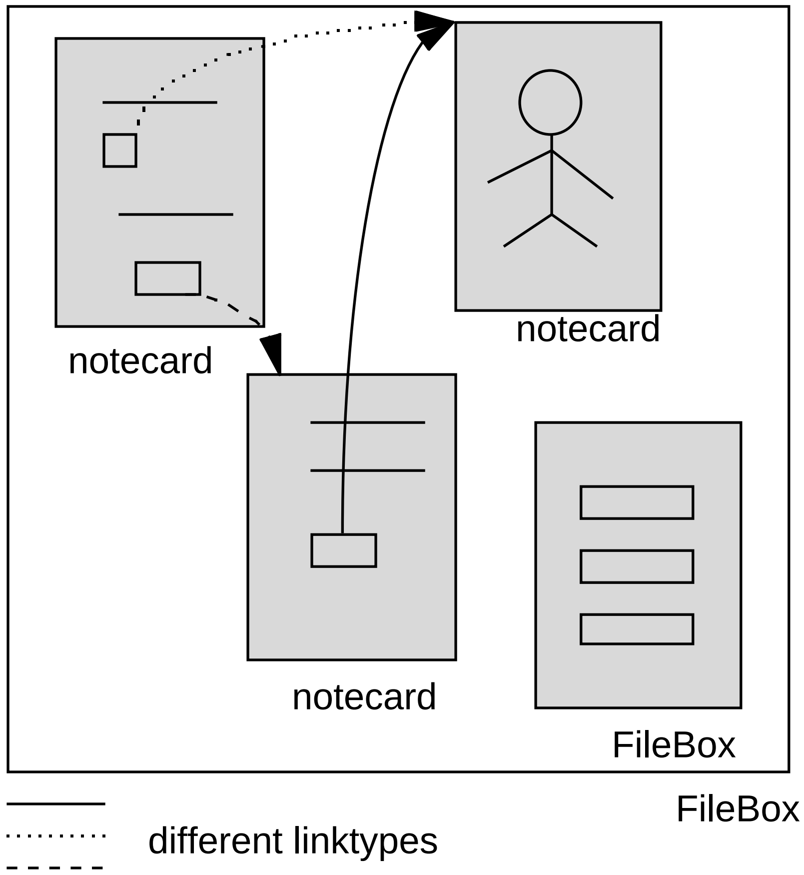
- **create** HAM object
- **delete** HAM object
- **destroy** HAM object
- **change** HAM object
- **get** data from HAM object
- **filter** information from a graph
- **special operations** (f.ex. searching string in node contents, merging contexts...)

The HAM storage model: Features

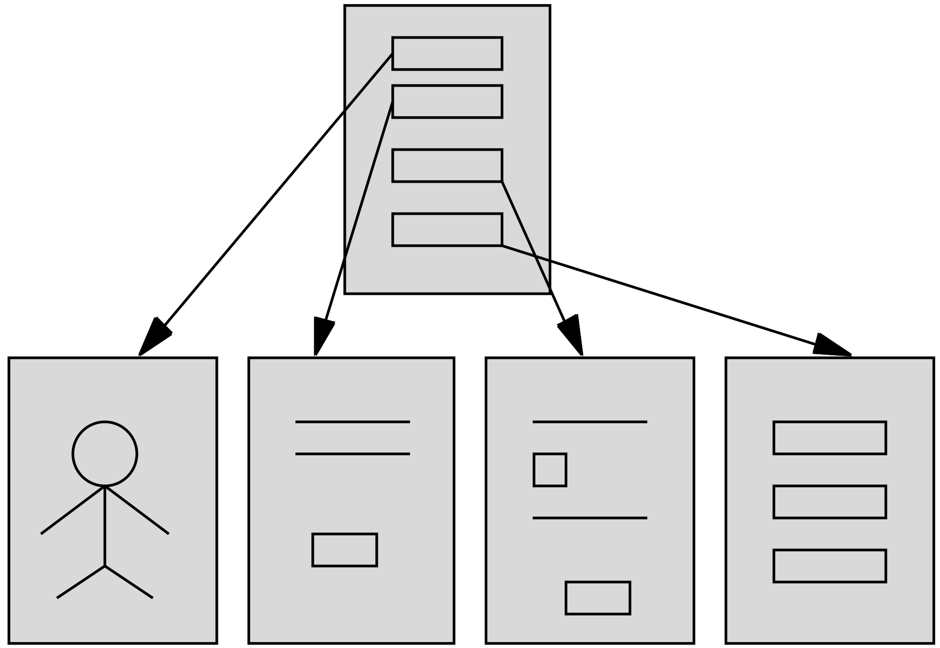
- Version History
  - The HAM provides an automatic version history mechansim.
  - Every object has a version time. Access to an object contains a version time.
  - A node can be classified as archived, non-archived and append-only.
- Filters - selective access
  - The user specify predicates and version time and the HAM retrieves the appropriate objects from the graph or smaller units.
- Data security - access restrictions
  - access control mechanism: ACL entry of an object consists of a user or group name and a set of permissions (access, annotate, update, destroy)

NoteCards: Concepts

semantic network composed of notecards
connected by typed links



file hierarchy



Example HAM application: NoteCards

- FileBox = HAM-node with the node attribute "*Filebox*"
- Notecard = HAM-node with the node attribute "*Notecard*"
- link to notecard = HAM-link with the link attribute "*Link to notecard*"
- link to FileBox = HAM-link with the link attribute "*Link to FileBox*"

HAM: Meeting Hypermedia requirements

Openness	**no**	**The server has to know about application-level abstractions.**
Extensibility and tailorability	**no**	
Versioning	**yes**	**provides simple time based versioning**
Search and Query	**yes**	**provides a simple query facility through filters**
Consistency **Virtual structures and computation**	**partly** **partly**	**data security mechanism** **provides a demon mechanism that invokes application or user code when a specific HAM event occurs**
Multimedia **Collaborative work/multiuser**	**no** **yes**	 **allows access restriction through a data security mechanism**
Composition	**partly**	**context concept**

simple and inflexible solutions

HAM and Hyperform

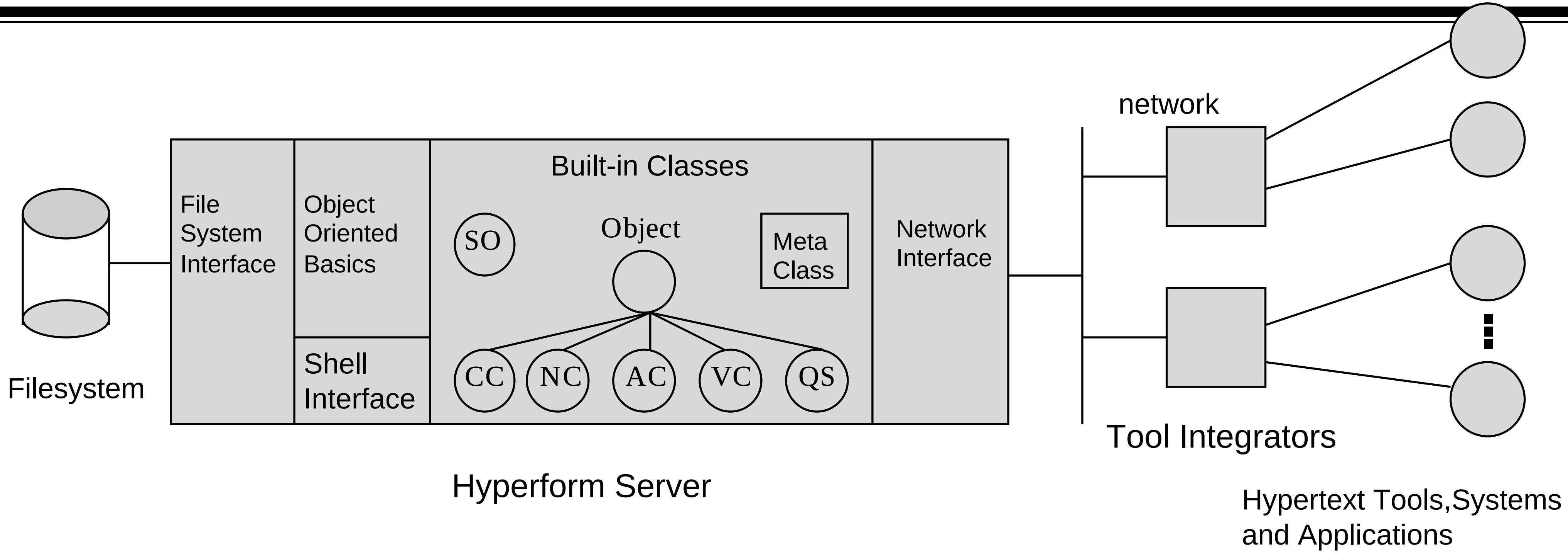
HAM, a HAM-generation HBMS

- gives the system developers the possibility to concentrate on the application and user interface layer
- monolithic and non-extensible
- provides fixed hyperbase support: *How do I make the best use of the services provided in the hyperbase?*
- The developer has to think in terms of the provided hyperbase support when designing the other layers.
- HAM, the first step towards a storage model standard?

Hyperform, a next generation HBMS

- non-monolithic and extensible
- provides flexible hyperbase support: *Which services would I like the hyperbase to provide?*
- Developers can avoid making undesirable design trade-offs due to fixed hyperbase support.

Hyperform: Architecture



Hyperform: Design goals

- providing an extensible application independent hyperbase service within an open, distributed architectural setting
- providing effective data management support for advanced hypertext environments
- reducing the effort required to develop high quality customized hyperbase support for distributed hypertext applications
- can be used for research for future hypertext applications
- provides a platform for adressing the major issues of the next generation of hypertext systems through a rapid prototyping approach
- to be a multi-hyperbase system
- future applications of Hyperform will help building a large library of useful hypertext data models and hyperbase configurations.

Hyperform: Concepts

- Basic object-oriented data modeling features (implemented in Meta Class and Object Class)
- Concurrency Control (implemented in CC class)
- Notification Control (implemented in NC class)
- Access Control (implemented in AC class)
- Version Control (implemented in VC class)
- Query and Search (implemented in QS class)

Hyperform: The object concept

- structure of instances/objects:

  (class-uid (read-only attributes)(read-write attributes)(calculated attributes)
  (dynamically allocated attributes))

- structure of classes:

  (anchestor descendant super-classes subclasses class-description)

- 5 basic methods are provided (get-instance, delete-instance, get-attribute, delete-attribute, set-attribute)
- message passing: (send object message .args)
- support for
  - object identity
  - encapsulation
  - object specialization
  - multiple inheritance
  - class evolution

Hyperform: Concurrency Control Object

Locking mechanism

- locking at the attribute and instance level
- locked attributes and instances can be read

Transaction mechanism

- dead-lock free transactions (2-phase locking protocol)

Hyperform does not introduce policies.

- The CC mechanism leaves it to the application programmer to decide which policy to use in each configuration of Hyperform.

Hyperform: Notification Control Object

Notification mechanism

- Users subscribe to events.
- Events can be any Scheme expression.
- Example: (event (lambda ()

  (if (and (= NC-entity 5)(equal? user "leggett"))

  NC-operation

  # f)))
- Send-events can be included in all methods in subclasses created in HF.
- Very flexible mechanism

Hyperform: Access Control Object

Object level access control

- three levels of object protection: get, set and delete
- notion of “user/group/other”

Annotation rights:

- are not part of the general services of the AC Object.
- The AC class has to be subclassed to implement annotation rights once a data model is loaded.

```
((class-name AC-object)
 (read-write-att (owner()) (created())
                 (modified-by()) (modified ())
                 (permission “gsdg__g__”)
     (group()))
(super-class object)
(methods
   (define (AC-init self . args) ...)
   (define (change-permission self perm) ...)
   (define (change-group self group) ...)
   (define (get-permission self) ...)
   (define (set-permission self) ...)
   (define (delete-permission self) ...)))
```

Hyperform: Version Control Object

Versioning data

- tree model
- version attributes: version, previous, next, variants
- support for revisions, variants and releases in delta and fully constituted form.

Versioning structure

- Done by subclassing the VC class once a data model is introduced.

Hyperform: Query and Search Object

<u>Content search</u>

- two variants of filter functions (matching values in attributes):
  - one for classes: match
  - one for instances: filter
- basic list operations (union, intersect, subtract, etc.)

<u>Structure search</u>

- no direct support
- can be built on top of the basic content search facilities once a data model is introduced, the Query and Search class has to be subclassed.

Hyperform: Extensibility and tailorability

The 5 subclasses should be further subclassed in order to extend and tailor the provided mechanism and to introduce the appropriate data model.

Hyperform: Applications

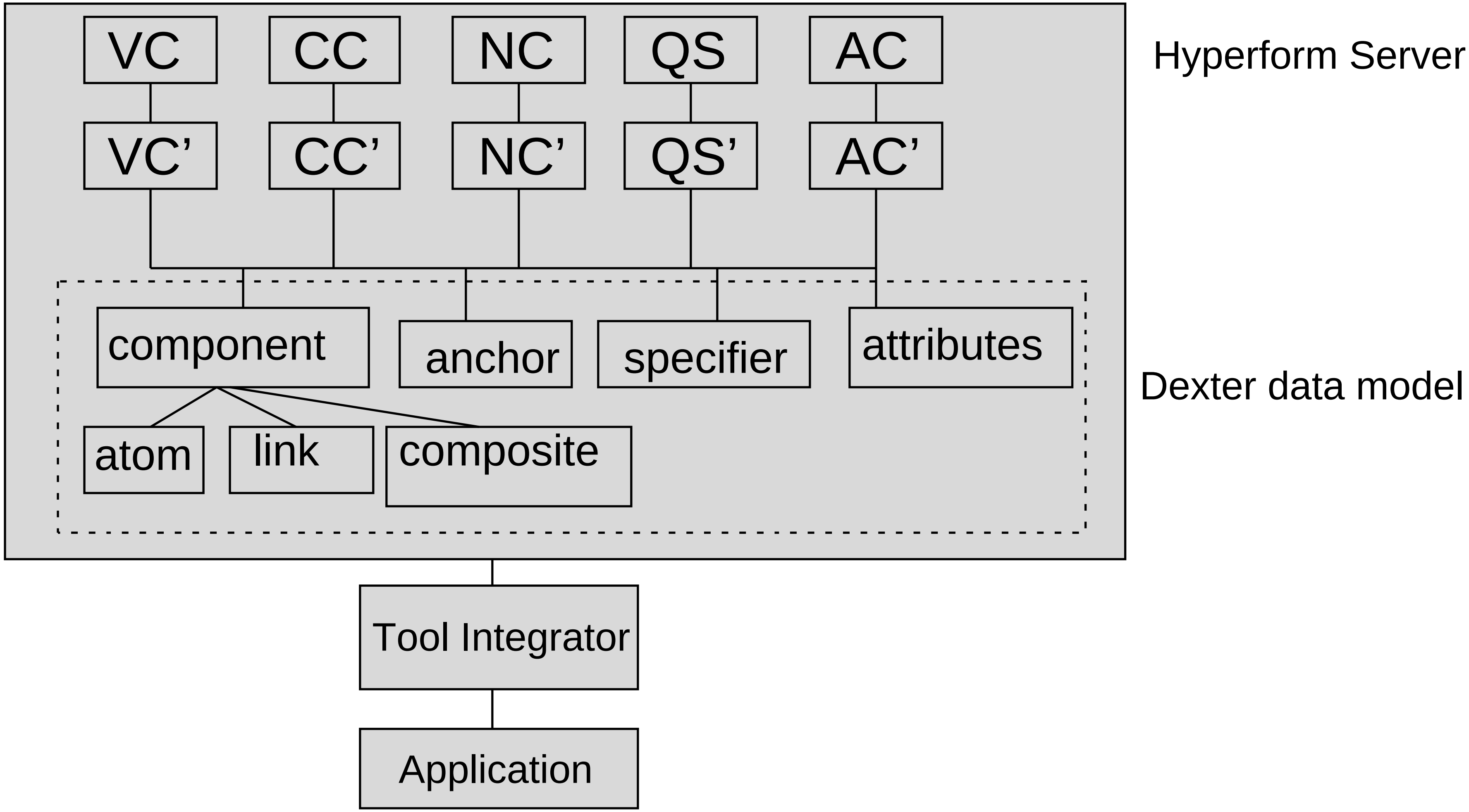
- simulation of existing HBMSs
- developing new HBMSs
- simulation of existing link engines
- data exchange between existing hypertext data models
- research engine for future hypertext applications

Hyperform: Applications

Steps for developing specific hyperbase configurations

- **Policy setting stage**
  - a data model must be determined
  - decisions concerning the degree of concurrency control, notification control, version control, access control and query and search support must be made
  -
- **Implementation stage**
  - create new versions of the 5 built-in classes in order to specialize (tailor, extend and evt. shadow) the provided application independent HBMS services of the 5 built-in classes to provide the desired functionality
  - implement the data model and HBMS configuration by creating new classes inheriting features from the 5 specialized classes

Hyperform: Developing a HBMS supporting the Dexter data model



Hyperform: Meeting Hypermedia requirements

Openness	**yes**	
Extensibility and tailorability	**yes**	**at two levels**
Versioning	**contents: yes** **structure: indirectly**	**supported by the VC object,** **versioning structure done by subclassing**
Search and Query **Consistency**	**content search: yes** **structure search: indirectly** **yes**	**supported by the QS object** **(provides content search, structure search can be built on top of the basic content search facilities once a data model is introduced)** **supported by the AC object**
Virtual structures and computation	**yes**	**Computation is supported both over and within the essential components of hypertext by calculated attributes and the event mechanism.**
Multimedia	**partly**	**possibility for distribution and handling of other medias**
Collaborative work/multiuser	**yes**	**supported by the CC object and the NC object**
Composition	**indirectly**	**provides the possibility to form any data model including composite objects**