Multimedia Databases

Motivation: the study of multimedia databases is influenced by various applications and their exponential expantion in the real life.

Concepts: What is Multimedia and Hypermedia? A lot of definitions (the definitions below were given by Marmann and by Steinmetz).

Multimedia (by Marmann):

A multimedia system is a computer controlled integration of medial information objects of different types (text, images, audio, video,). The integration refers to:

- Data modeling
- Storage
- Presentation
- 'I'ime synchronization

A promise is that the media must be digitally represented, or at least digitally controllable.

Multimedia (by Steinmetz):

A multimedia system is to be defined as computer controlled, integrated

- generation,
- manipulation
- representation and
- communication

of independent information. This information is coded in at least one continuous (time dependent) and one discrete (time independent) medium.

Hypertext:

Text, that is not linear, but has a net structure with nodes and links. Link sources and link targets are called anchors.

Hypermedia:

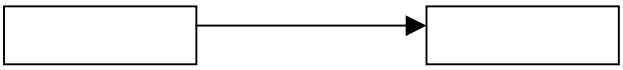
Nodes can contain any arbitrary media, not only text. This means that the linka from and to time dependent media are possible.

Link concepts:

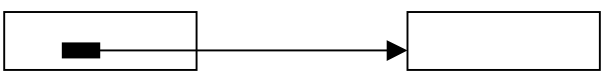
Links may be classified by various attributes. These include granularity, direction, functionality, locality, representation and dynamics of links.

1. **Anchor granularity:**

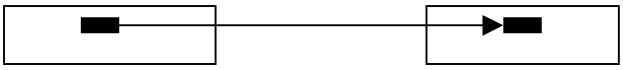
Node-to-Node-Links. They only allow complete hypertext nodcs as source and target nodes.



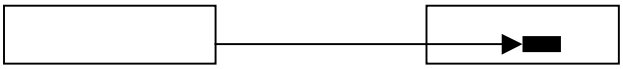
Span-to-Node-Links. They point from a single, free definable area within a node (a span) to a complete node.



Span-toSpan-Links. They admit spans within nodes as both, link source and link target.



Node-to-Span-Links. They point from a whole node to a span (used very rarely).



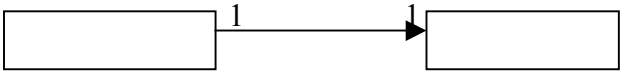
2. Direction of links:

onedirectional links: can be followed only from the source to the target

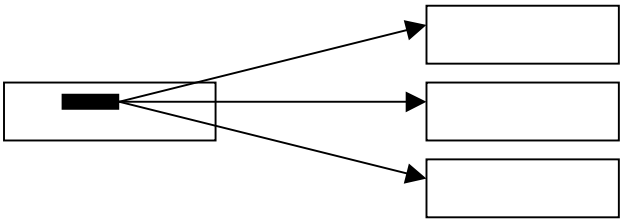
bidirectional links: **can be followed both ways**

3. Functionality of links:

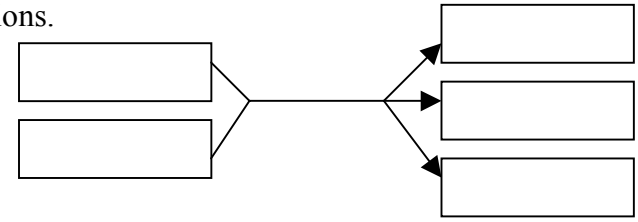
- 1:1 - link – only one source and one destination anchor (a clssical form of links)



- 1:M - link – starting with one source, several destinations can be reached

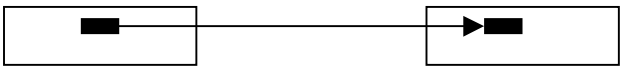


- N:M - link or MSMD-link (Multi-Source-Multi-Destination-Link): it has several sources and several destinations.

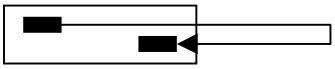


4. Locality of links:

- inter-document link



- intra-document link



5. Representation of links:

- implicit, i.e. the link does not exist as an entity of its own, but is defined implicitely in the document. (Example: in WWW, the link target is specified directly in the HTML document within the link source anchor)

- explicit, i.e. the link exists as an entity of its own, and attributes can be accorded to it. These attributes specify the link, e.g. author or type. In the document there exist only anchor-definitions. The link entity refers to these anchor-IDs.

6. Dynamic of links:

static (stored link), i.e. the link is persistently stored in the system (implicit or explicit).
dynamic (computer link), i.e. the link (resp. the link target) is computed during runtime. The system does not contain any specifIcation on the link. Dynamic links are uscd very often in WWW for database search on cgi-scripts (also compare to links in digital library systems including full text search; sometimes known in the literature as content based search).

Features of Multimedia Database Systems

- The multimedia database systems are to be used when it is required to administrate a huge amounts of multimedia data objects of different types of data media (optical storage, video tapes, audio records, etc.) so that they can be used (that is, efficiently accessed and searched) for as many applications as needed.
- The Objects of Multimedia Data are: text, images. graphics, sound recordings, video recordings, signals, etc., that are digitalized and stored.
- Multimedia Data are to be compared in the following way:

Medium	Elements	Configuration	Typical size	Time dependent	Sense
Text	Printable characters	Sequence	10 KB (5 pages)	no	visuall/ acoustic
Graphic	Vectors, regions	Set	10 KB	no	visuall
Raster image	Pixels	Matrix	1 MB	no	visuall
Audio	Sound/volume	Sequence	600 MB (AudioCD)	yes	acoustic
Video-Clip	Raster image/ graphics	Sequence	2 GB (30 min.)	yes	visuall

The need and efficiency of MM-DBS are to be defined by following requirements:

Basic service:

- to be used for multiple applications
- not applicable as a real end-user system (like program interface)

Storage and retrieval of MM-Data:

For the Storage:

- input of MM objects
- cmnposition (to multimedia objects) (example: authoring systems)
- archive of data (in hardware and format independent way)

For the Retrieval:

- support of complex search
- efficiency (indices etc.)
- evaluation (aggregation, filtering)
- preview
- also conversions (needed to gain or lead to hardware and format independence)

For the Update

- only replace or also edit? (the complexity depends on).

Multimedia Database Systems have to be capable:

1. Support of multimedia data types, i.e. data types as data structures, including type of data and operations

2. Capability to manage very numerous multimedia objects, store them and search for them

3. To include a suitable memory management system, to improve performance, high capacity, cost optimization

4. Database system features:
   - persistency
   - transaction concept
   - multi-user capability
   - recovery
   - ad-hoc queries
   - integrity constrains (which leads to cocsistency)
   - safety
   - performance

5. Information retrieval features:
   - attribute-based search
   - content-based search

Integrity Constrains for MM-DB Applications

The following features are typical for MMDB:

- Unique, Primary-key Constraints
- Referential integrity
  - via foreign keys (RM)
  - via OIDs (OO)
- Existential integrity
- NOT NULL constraints
- Integrity rules (check clauses)
- Trigger

Specifically for OO:

- Pre- and postconditions for methods
- Constraints of the class hierarchy
- Partition conditions (Disjointness constraints)

The Dexter Reference Model

The Dexter Hypertext Reference Model

The goal of reference models is to built a common basis for the architectures of hypertext systems, so that can interact to and to be compared to each other.

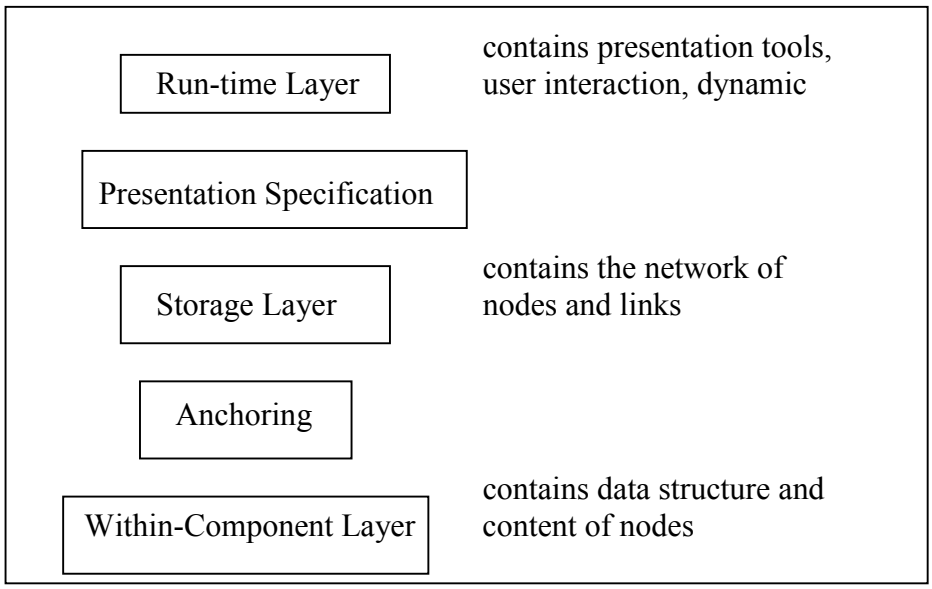
Objectives for the model:

- Standard terminology
- Minimum functionality
- Reference model – a meta-model for the formal description of properties of (existing) hypermedia systems
- Basis for the development of exchange formats.

The results are also applicable to hypermedia systems.

An Overview on the Dexter Model:

It consists of three layers and two interfaces



The three layers:

- The **Storage Layer** describes the network of nodes and links (most important layer). The storable Hypermedia objects are called components. The three basic components arc:
  - Atom
  - Link
  - Composite Component (for (hierarchical) structuring)

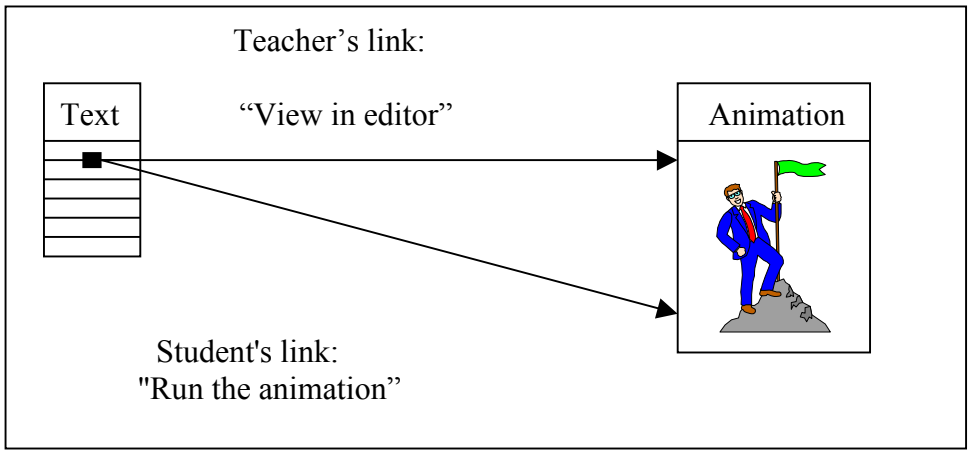
- The **Within-Component Layer** describes the content and structure of the components (nodes, links); e.g. the data structure for text, images, animation etc. This layer is system specific, so it will not be defined in the Dexter Model (e.g. defined in ODA, IGES, etc.)

- **The Run-time Layer** manages the presentation of the components in the user interface at runtime. There will be administered one session per user. This layer also handles the read/write-copies of the components in the cache (because updates are possible).

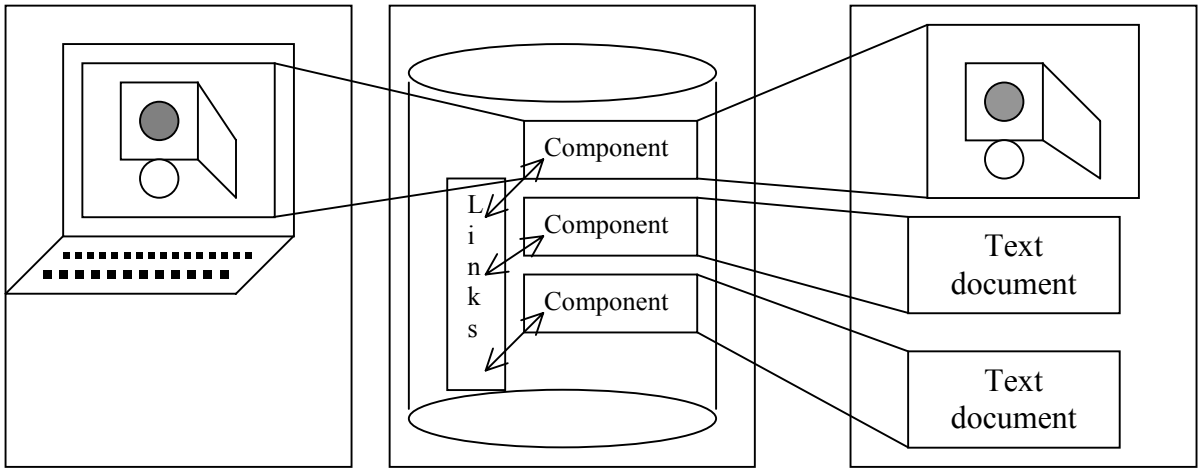
The two interfaces:

- The Anchoring offers the mechanism for addressing the link sources and targets even inside of the components (Span-to-Span-Links).

Thee Presentation Specification offers a possibility to deposit information on the display of components in the Storage Layer by the Run-Time-Layer (e.g. editable/noneditable)



Example for presentation specification even on links.



The Three Layers in Detail

The Storage Layer

This layer describes the hypertext structure as a finite set of components and a set of two access functions: resolver- and accessor- function.

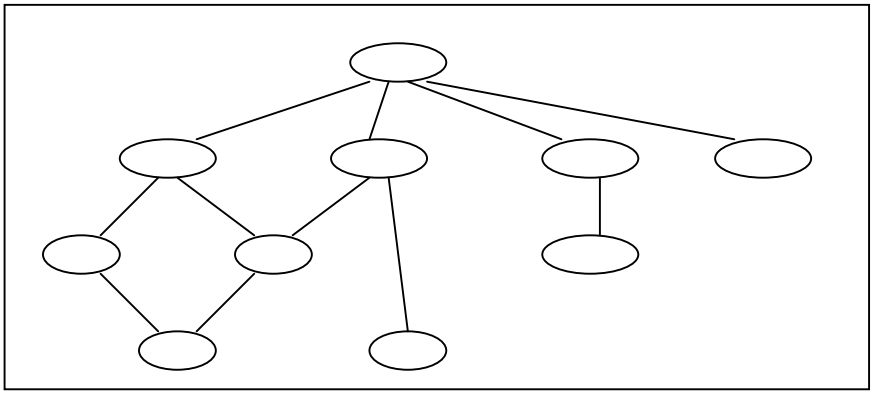
Component = Atom
Link
Composit Component

Atom: is cobsidered as a "node", that means as a primitive, it's substructure and content are specified in the Within-Component Layer.

Link: is specified by two or more anchors (= endpoints). Anchors can be components or parts of components (Span-to-Span).

Composite Component: it is a hierarchical structuring of components, particularly: DAG

Identification: Each component has a "global unique identifier" (UID).



The two access functions on components:

accessor-function: UlD → component

resolver-function: predicative spezification of the link target → UIDs or Ω

Anchors:

Span-to-Span-Links must be able to address substructures of components.

Anchor = (anchor-id, anchor-Value)

locally unique within a component (fixed length)

specifies location inside the component (variable length)

globally unique link anchor: Tuple (UID, anchor-id)

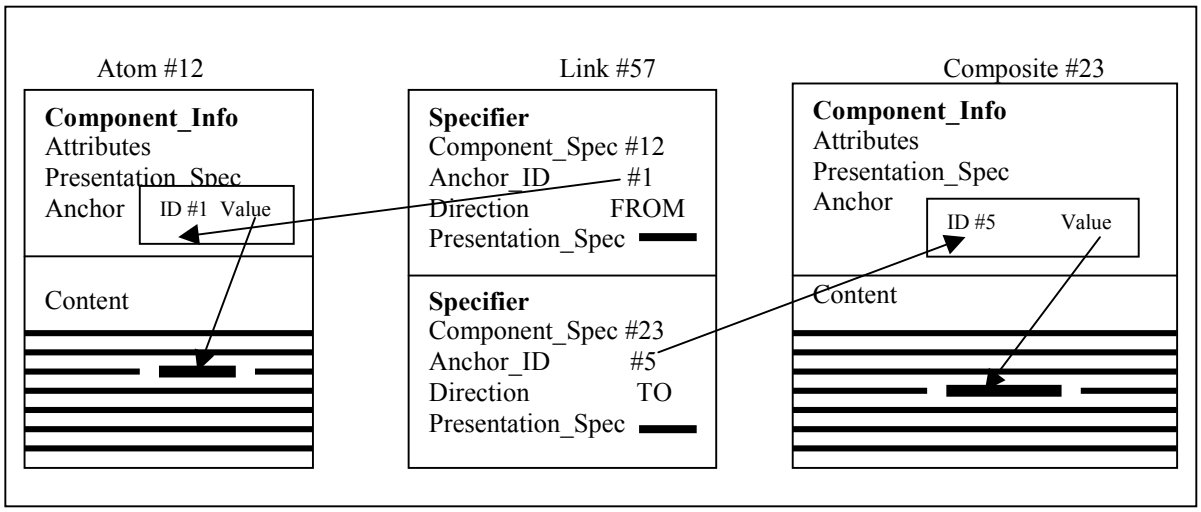
Note:

According to the definition, links are components. Thus the following are possible: Link with endpoints that are links, but also: endpoints of a link inside another link. Example: link on a linkdescription or on the type or direction of another link.

Specifier: of possible link endpoints; a specifier consists of

- component specification (e.g. the UID)
- anchor-id (= AID)
- direction (optional) ∈ [FROM, TO, BIDIRECT, NONE]
- presentation specification (optional)

Link: sequence of two or more specifiers (1:1-, 1:M-, M:N-links). Mostly: 1:1 links directed links

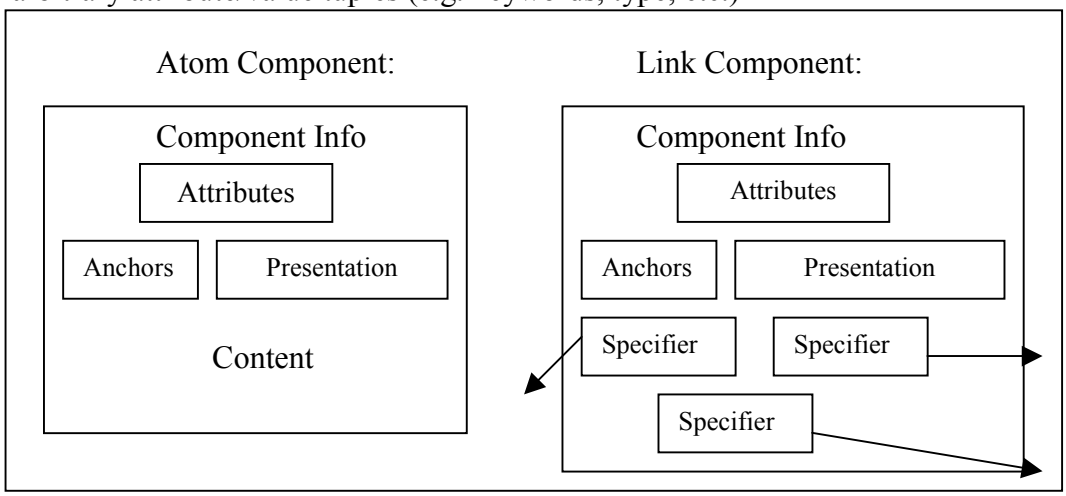


Component-Info:

In addition, components have the socalled "component informations".

These do not describe the content, but the properties of a component:

- all the anchors
- presentation information for the Run-time Layer
- arbitrary attribute/value tuples (e.g. keywords, type, etc.)



Components Structure in the Storage Layer

Link-Consistency:

The component-specification of each link-specifier must lead to an existing component (resolve, access).

Links are entities of their own, thus deleting a component requires deletion or update of all the corresponding links.

The Within-Component Layer:

This layer contains the data structure of the component and their contents. It is system specific, so it is not defined any closer in the Dexter Model.

The Run-time Layer

Runtime system for presentation. It contains plenty of dynamic functions, like "present component", "follow link", "realize", etc.

A new session is started for every user (session management).

Instance of a component = the users representation of the component
More precise: it is the copy of the component in the run time cache (compare to modification and rewrite), i.e. several instances of a component may exist at the same time. Each instance has it's own instance identifier (IID).

The instantiation of a component necessitates the instantiation of the anchors (= link markers).

Session-Entity contains:
- the hypertext
-the mapping IlDs -> UlDs
-History (at the moment this is defined only for read-only;

writes: compare to the TA-concept in DBS)
-runtime resolver function: (specification -> UlDs)
-instantiator function: (UID + presentation-spec. -> IID + Instance)
-realizer function: (invcrs to instantiator, = write back the cache after a modification)

Follow-Link operation: input: IID of an instance and the Link Marker.

Example for a simple exchange format according to the Dexter Model:

```
<hypertext>
    <component>
        <uid> 21 </uid>
        <type> text </type>
        <anchor>
            <id> I </id>
            <location> d13 </location>
        </anchor>
        <data> This is some text ... </data>
    </component>
    <component>
        <uid> 777 </uid>
        <type> text </type>
        <anchor>
            <id> I </id>
            <location> 13-19 </location>
        </anchor>
        <data> This is some other text ... </data>
    </component>
    <component>
        <uid> 881 </uid>
        <type> link </type>
        <specifier>
            <component_uid> 21 </component_uid>
            <anchor_id> 1 </anchor_id>
            <direction> FROM </direction>
        </specifier>
        <specifier>
            <component_uid> 777 </component_uid>
            <anchor_id> 1 </anchor_id>
            <direction> TO </direction>
        </specifier>
    </component>
</hypertext>
```

Evaluation and Review

The Dexter Model is a more powerful model then the models that are usually used in Hypemedia systems:
-1:M links, M:N-Links
-Links to links (hard to implement)
-Complex components
-bidirectional links
-extremely powerfull resolver-function

Weak Points of the Model:

- strict separation of the layers is weakened by the anchors!
- the direction of a link is not specified until in the Link Specifier; thus inconsistency problems may occur;
- model extensions will be necessary in respect of time-critical media (synchronisation, etc.) (see also: Amsterdam Hypermedia Model).
- in editing mode: all relevant components must already exist in the system, before a link can be defined.

Summary:

The Dexter Model is not a standard in it's true sense of the meaning, (not in the sense: "We fulfill the Dexter-Model"), but a reference model (instead of: "Our ... corresponds to the Dexter Model").

Advantages of Multimedia Database Systems

- integrated administration of huge amounts of multimedia data
- optimized storage
- efficient access
- manyfold complex search possibilities
- referential integrity of links
- transaction protected multiuser mode
- recovery
- etc.

Multimedia-DB applications

Fields of application:

- *static/passiv:*

Retrieval / Information / Archive
(Libraries, video on demand, information systems, press, hospitals)
Databases, information retrieval

- *static/aktiv*

Education / Commercials/ Entertainment
(School, university, professional training, games, commercials)
CSE, Teachware, Courseware, CBT,

- *dynamic/passiv*

Writing / Publications/ Design
(Press, engineering, architecture)
Editors, layout generators, CAD-systems

- *dynamic/active*

Controlling/Monitoring (Factories, traffic, weatherforecast, military)
Process control systems