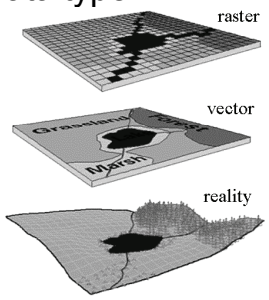
Management of Multidimensional Discrete Data

Image data type

- Vector data
- Raster data
- Transformation



- Multidimensional discrete Data
- Traditional database support of MDD
  - Store in linear byte stream
  - Lack of data independence
  - No operations
- Requirements of MDD database
  - Structure definition of arrays and operations on arbitrary pixel types
  - Efficient access methods and compression mechanisms

Image modeling of MDD - operations

- Schema-level operations
  - Based on fixed array base type
  - E.g. Querying the color space
- Instance-level operations

Image modeling of MDD – operations (cont.)

- Instance-level operations
  - Image range
  - Image restriction
    - Reduce client memory space and computation work overhead
  - Unary and binary operations
  - Image extension
  - Template
    - Full power of template
    - Specialized template operations: smooth, extract, flip

Image modeling of MDD – data access

- The pixel traversal sequence
  - Irrelevant
  - Relevant, known in advance
  - Not predictable

Formal Semantics for MDD Definition and Manipulation

- Value semantics
  - Constants
  - Trimming and projection
  - Induced operations
  - Predicate iterators
  - What about other traditional database operations? (aggregations?)

Formal Semantics for MDD Definition and Manipulation (cont.)

- Update Semantics
  - Fixed array base type and array dimension
  - Initialization
  - Assignment
  - Partial assignment

MDD Definition and Manipulation

- TDL of APRIL (ooDMBS)
- Object constituents specified in object type definition
  - Set of attributes
  - Object contents (long field)
  - Successor clause

MDD Structure Definition

- Embedding of MDD into the APRIL
  - Overlaying the contents string with a structure definition
    - Formal structure definition:
      - typedef B T [r1]...[rd];
- example:
  - Typedef unsigned int GrayscaleMatrix [640][480];

```
typedef object
{     String description;
      unsigned int contents [640] [480];
}GrayscaleImage;
```

MDD Query Language

- APRIL Query Language
  - Select <object type> where <condition>
- Required language extensions for querying and updating MDD data
  - Constants
  - Array Manipulator
  - Induced Operations
  - Predicate Iterators
  - Update Semantics

Constants

- Small arrays: {e1,...,en} | expressions of type T
  - {{1,0,-1}
  - {2,0,-2} : Sobel filter template
  - {1,0,-1}}
- Large arrays: {e: [r1,...rd]}
  - {0:[1024, 768]} : black image of size 1024 by 768

Array Manipulator

- Combination of trimming and projection
  - a[r1,...rd]
- Example: The first 40 pixel lines of all G3 telefaxes
  - Select G3Fax.contents[#, 0 .. 39]

Induced Operations

- Binary induced operations
  - Addition: a, b are RGB images
    - a+b = component-wise addition on RGB integer triplets {R(a)+R(b), G(a)+G(b), B(a)+B(b)}

Predicate Iterators

- "all" and "some"
- example
  - Select GrayScaleImage
  - where all(case when *m* then
    - GrayScaleImage.contents > *t*
    - else true end)

Update Semantics

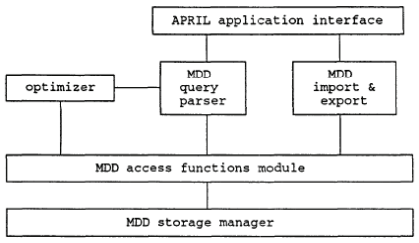
- Total update
  - update T set a=v
- Partial update
  - update T set a[r1,...rd] = v
- example

```
update Artwork
set    Artwork.contents[
  range1(Artwork.contents)-range1(s) .. range1(Artwork.contents),
  0 .. range2(s)] = s
```

MDD Management

- Tiling: rectangular cut-out of an MDD object obtained through trim operations,
  - Data are stored sequentially inside a tile
  - Store and load as a whole
- Spatial indexing
  - Access to tiles

Architecture of MDD Management Subsytsem



Query Transformation

- Transform query expressions into canonical forms;
  - trimming and projection is pushed down
  - induced operations are combined
- Transformation rules: trim commutativity, trim absorption,induced function pullout, etc.

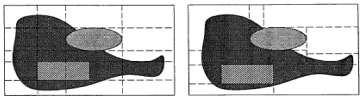
Query Evaluation

- Create a trim list from the canonical expression
  - for each $\mathtt{trim}_{[i,k]}$ in the chain, tl contains a triple (i,u),
  - for each $\mathtt{proj}_{[p]}$ in the chain, tl contains a triple (p,r),
  - for each i with $1 \le i \le d$ where neither $\mathtt{trim}_{[i,k]}$ nor $\mathtt{proj}_{[i]}$ is in the chain, tl contains a triple $(i,-\infty,+\infty)$.
- Find tiles associated with trims in the list
- Optimize if possible (rearrange the *tile list*)
- Fetch the tile and extract required part

Tile Indexing

- Tiles (oid: raw, **tid: number**, comp:char, bucket: long raw)
- Index requirement depends on tiling strategy
  - Regular gridding → no index is needed
  - Irregular gridding → materialized, spatial index for tile lookup

Tile Indexing (cont.)



Q: Why does it support various strategies?

Index Management

- Finite MDD range
- Natural bounding boxes for spatial index method
- Not overlapping tiles
- Infrequent index updates
- Index construction in one transaction

Summary

- Goal: Extending a DMBS with support for MDD
- Requirements taken from Image Algebra: some set of operations
- Sublanguage for MDD manipulation
- Storage manager: tiles +spatial index

Thanks!