Management of Multidimensional Discrete Data

Image data type
- Vector data
- Raster data
- Transformation

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 range
  - Image restriction
  - Image extension
  - Template
  - The pixel traversal sequence
    - Irrelevant
    - Relevant, known in advance
    - Not predictable

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
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:
      

\[
\text{typedef} \ B \ T \ [r1]...[rd];
\]

- Example:
  
\[
\text{typedef unsigned int GrayscaleMatrix [640][480];}
\]

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
  - \( a[r_1,...,r_d] \)
- 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 \( a[r_1,...,r_d] = v \)
- Example
  - Update \( T \) set \( a[r_1,...,r_d] = v \)

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 Subsytem
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 trim \( t \in T \), if \( t \) contains a triple \( (A, v, w) \)
  - for each \( p \in P \), if \( p \) contains a triple \( (A, v, w) \)
  - for each \( T \) with \( T \in T \) when \( t \in T \) and \( t \) contains a triple \( (A, v, w) \)
- 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 (\( \text{oid: raw, tid: number, comp:char, bucket: long raw} \))
- Index requirement depends on tiling strategy
  - Regular gridding \( \rightarrow \) no index needed
  - Irregular gridding \( \rightarrow \) materialized, spatial index for tile lookup

Index Management

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

Tile Indexing (cont.)

- Q: Why does it support various strategies?

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!