An Array Library for MS SQL Server

Scientific requirements and an implementation

László Dobos\textsuperscript{1,2} -- dobos@complex.elte.hu
Alex Szalay\textsuperscript{2}, José Blakeley\textsuperscript{3},
Tamás Budavári\textsuperscript{2}, István Csabai\textsuperscript{1,2}, Dragan Tomic\textsuperscript{3},
Milos Milovanovic\textsuperscript{3}, Marko Tintor\textsuperscript{3}, Andrija Jovanovic\textsuperscript{3}

\textsuperscript{1}Eötvös University, Budapest, Hungary
Department of Physics of Complex Systems
\textsuperscript{2}The Johns Hopkins University, Baltimore, USA
\textsuperscript{3}Microsoft
Outline

• A short review of our earlier projects
• Scientific motivations and some use cases
• Requirements for a simple array type
• Pros and cons of using RDBMS systems
• An implementation using MSSQL
• Possible upgrades to MSSQL
• Conclusions and Summary
Astronomical DBs in the exponential era

- Systematic surveying of the sky
- Repeated observations
  - Fainter objects
  - Time domain data
- Multiple wavelengths
  - Different data sets
  - Match by coordinates
- New software tools are needed
Earlier projects: SkyServer

- MSSQL database
- Astronomical data from the Sloan Digital Sky Survey (SDSS)
- Brightness measurements for 580 million objects: stars, galaxies, etc. in five color filters
- 4.5 TB row data to date (+ images in blobs)
- Introduced SQL to the astronomer community
  - Minimal GUI
  - Main interface is pure SQL
  - Lots of user-defined functions
  - Users use ‘MyDB’ to store results
  - Spatial search capabilities

http://skyserver.sdss.org
Alex Szalay, Jim Gray et al.
Our earlier projects: Spectrum Services

- Sloan Digital Sky Survey spectroscopic observations
- 1 million spectra
- 1 TB range
- Spectra are high dimensional vectors in blobs ($D > 3000$)
- Rich web based GUI + Web services
- User built pipelines for reprocessing measurements to fit science needs
- Search based on spectral features

http://voservices.net/spectrum
Dobos et al.
Astronomy

- **Living data**
  - Recalibration, transformation on daily basis
  - Large bulk-inserts on weekly basis, merging

- **Observations of individual objects**
  - Point-like data ($D = 5 \ldots 10$)
  - On the order of billions of rows
  - Automatic classification, cluster finding algorithms

- **Spatial search**
  - High precision spherical geometry (milli-arcsecond resolution)
  - Tree based search in many dimensions

- **2-point and n-point correlation functions**
  - On the sphere and in many dimensions
Astronomy and astrophysics

- **Spectroscopy – o(100 GB) range**
  - High dimensional vectors \([D = o(1000)]\)
  - Complete reruns of data processing on daily basis
  - Resampling, PCA, function fitting, etc.

- **Cosmological simulations – o(100 TB)**
  - Position and velocity for each particle
    - No grid, high number of particles \([N_p = o(10^6 - 10^7)]\)
    - Data dumps at timesteps \([N_t = o(10^3)]\)
    - Multiple runs of the simulation \([N_r = 500]\)
  - Individual points are impossible to store
  - Organize points in octree, store chunks of data in arrays
  - Identify clusters of particles – friends-of-friends algorithm
  - Compute density over regular grid, FFT, correlation functions

Image from: Springel, V. et al. 2005, Nature

EDBT/ICDT 2011 Uppsala, Sweden – Workshop on Array Databases
Turbulent flow simulations

- Distribution of data over a regular grid [size o(100 TB)]
- $1024^3$ grid, 2000 time steps, velocity, pressure [$D = 4$]
- Idea:
  - Do not download full data for analysis
  - Put virtual sensors into the flow (list of positions, time)
  - Output: velocity field at the given positions
- Store grid in chunks, with buffer zones
- Organize chunks along space-filling curve
- Interpolation using various kernels (hence the buffers)
- Visualization…


EDBT/ICDT 2011 Uppsala, Sweden – Workshop on Array Databases
Requirements for a SQL array type

• Simple extension to SQL
• Multi dimensions up to at least D = 6
• Main numerical data-types, complex numbers
• Interface to math libraries
  – LAPACK, FFTW, Matlab…
• High-performance subsetting functions
• Easy aggregation over arrays
• Efficient support for both
  – small arrays (in-page) and
  – large arrays (blob) data
Extending Microsoft SQL Server

• Pros:
  – Effective CPU, memory, disk and network management out of the box
  – Well supported by Microsoft
  – Extensibility via user-defined functions and types
  – .Net interface

• Drawbacks:
  – No native array support in SQL CLR
  – SQL CLR still lacks some features to be suitable
  – No access to original codebase
Our implementation

• Support for in-page and out-of-page arrays
• Implemented in .NET SQL CLR
  – C++/CLI for two reasons:
    • Benefit from template classes (no such in C#) to implement support for all base data types
    • Test possibilities
• Store data as pure binary with a short header
• No user-defined data types
  – Lots of user-defined functions instead
• Byte order to match common math libraries
Our implementation

- No SQL language extensions (yet)
- Sample code (kind of ugly), e.g.
  - Allocate vector with 5 items, get first item
    
    ```sql
    DECLARE @a VARBINARY(100) = FloatArray.Vector_5(1.0, 2.0, 3.0, 4.0, 5.0)
    SELECT FloatArray.Item_1(@a, 3)
    ```

  - Get subarray from another array @b
    
    ```sql
    SET @b = FloatArrayMax.Subarray(@b,
    IntArray.Vector_3(1, 4, 6),
    IntArray.Vector_3(5, 5, 5), 0)
    ```
Our implementation: Performance

• CPU limited performance with small arrays
  – Very fast I/O subsystem was used

• Performance mostly limited by the current capabilities of SQL CLR
  – Relatively expensive function calls
  – No cross-call caching of data
  – Data copy from native to managed memory required

• No numbers on out-of-page arrays yet
Possible upgrades to SQL CLR

• We used scalar functions because of UDT limitations

• To make UDTs suitable one needs
  – Partial Serialization/Deserialization of UDTs
  – Cross-row UDT instance caching
  – Support for ordered aggregation of UDTs
Conclusions and summary

• SQL is a must for astronomers these days

• Building a powerful array library in .NET SQL CLR is already feasible, though tricky

• Upgrades to user-defined types would be great

• SQL language extensions required
  – At least in a form of a pre-parser