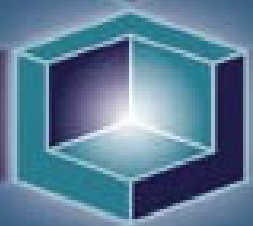
SmartArrays®

Array Technology for Business Analytics

Data Intensive Analytics



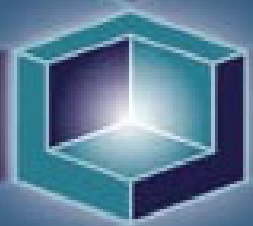
James Wheeler

SmartArrays®

Array Technology for Business Analytics

Data Intensive Analytics

- A Fresh Approach to Analytics with Array Technology
- Case Study: BNP Paribas
- Case Study: Office of Secretary of Defense
- How It works: Application Architecture
- Working with SmartArrays



Classic Analytic Technology

Technology	Issues
Database	Performance on large-scale analysis
OLAP / Multi-Dim Database	Expensive, restrictive,
Special Languages and math tools: K, A+, APL, Matlab…	Not suitable for integrated, production deployment
Build From Scratch	Time/cost to develop

The Missing Piece

Technology that

- Fits into enterprise development platforms: J2EE, .NET
- Does not disrupt existing systems
- Simplifies and accelerates development
- Promotes agile, bug-free development
- Handles huge data sets
- Really, really fast….

A Fresh Approach to Analytics

Most DBMS and OLAP tools were designed when computers were

- much slower,
- much smaller,
- much more expensive.

How can we tap the full power of today's big, fast, multi-gigabyte machines?

Our Approach: Array Technology

- Large, memory-resident arrays
- Suite of array-oriented data operations.
- Not just calculations: high-speed operations that
  - restructure,
  - compare,
  - filter,
  - sort,
  - combine data.

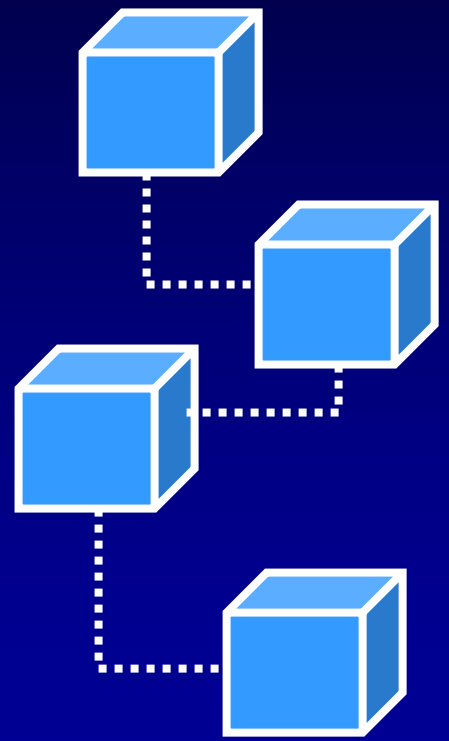
Our Approach: Array Technology

SmartArrays provides:

- A software toolkit for rapidly assembling fast analytic computations on bulk data.

Uses:

- Custom server-side “analytic middleware”
- Client-side analytics backed by central databases.



A Wealth of Array Techniques

- Bitmaps
- Multi-dimensional arrays
- “Whole array at once” arithmetic
- Sparse cubes
- Optimized sort and search functions
- Subscriptable ordered-memory arrays
- All functions work on all arrays – one array class holds all types of data.
- More than 250 operations in all

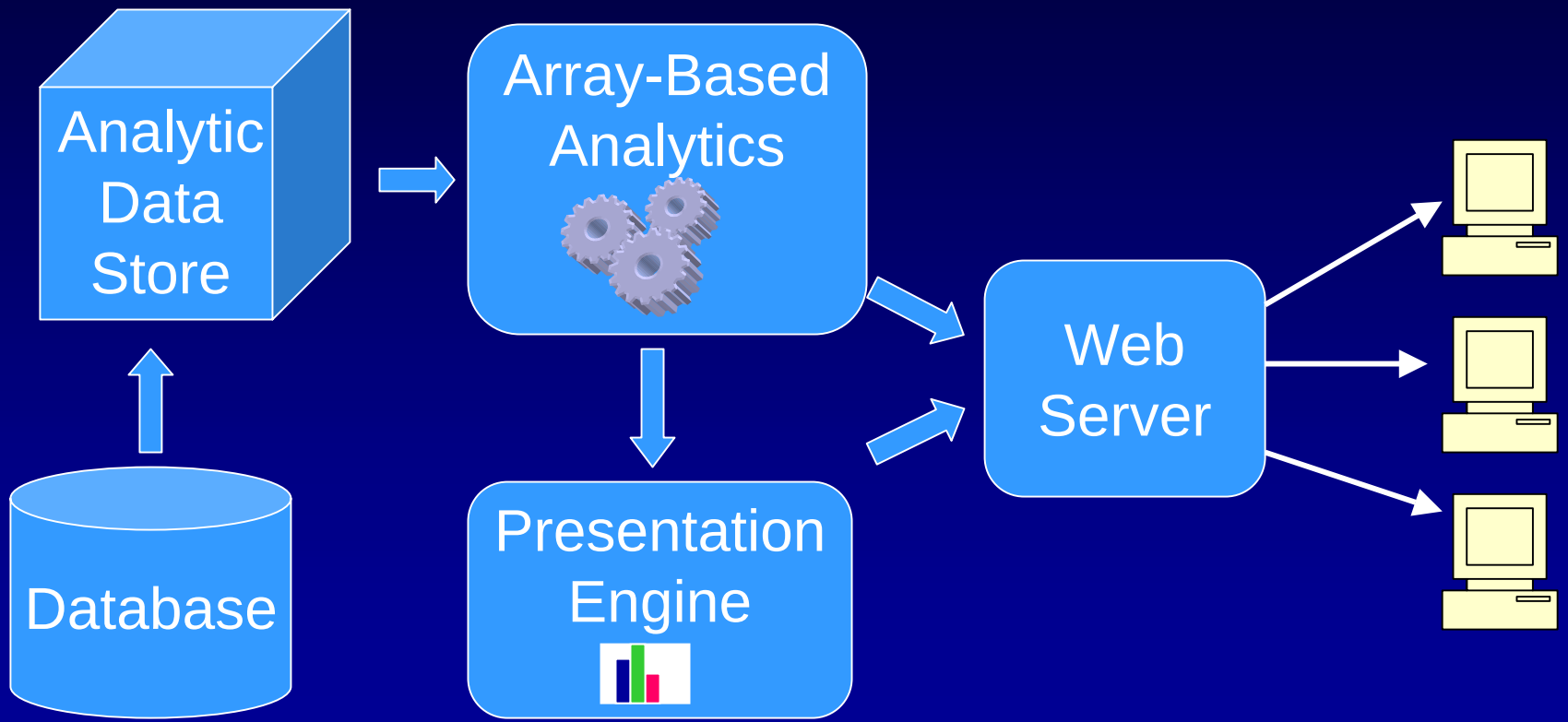
Goals of Analytic Solutions

- Web access (often)
- Fresh and comprehensive data
- User-formulated queries
- Instant results – while you remember the question
- Progressive knowledge discovery – each answer drives new question
- Drill-down to underlying details

What Does an Analytic Application Do?

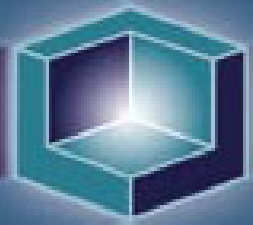
- Import data – various sources, formats
- Transform data – restructure, scale, map
- Hold data – for shared online access
- “Dimensionalize” – attributes of interest
- Slice subsets – per user specified filters
- Analyze – calculate, cross-tabulate, find trends
- Present – results in understandable form

Data-Intensive Web Analytics



SmartArrays®

Case Study: BNP Paribas



Case Study – BNP Paribas

Transaction Reconciliation System.

- Existing system written in Java with heavy reliance on database operations
- Performance issue: taking too much time to complete reconciliation
- Cocking & Co (London) engaged to try to improve throughput

Case Study – BNP Paribas

Cocking & Co's Solution: SmartRec

- Configurable reconciliation engine, written in Java using SmartArrays
- Data loaded into SmartArrays-based tables before analysis
- Simple declarative definition drives code to extract, transform data and reconcile records.

Case Study – BNP Paribas

First Results

- Test case ran 15 hours on busy server, estimated 90 minutes CPU time
- SmartRec processed same task in 80 seconds, on a laptop
- BNPP astonished and delighted with results

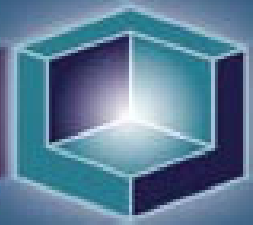
Case Study – BNP Paribas

SmartRec: Return on Investment

- Project cost justified by hardware savings alone
- Replaces existing custom solution
- Eventually will replace costly 3rd-party packages
- BNPP considering licensing to other firms

SmartArrays®

Case Study: Office of the Secretary of Defense



Case Study – Office of the Secretary of Defense

Problem:

- Huge database of processing history on 50,000,000 DoD requisitions
- Goal: Set performance criteria based on measuring past results
- Track and measure improvements in how fast orders are processed.

Case Study – Office of the Secretary of Defense

Technical Challenges:

- Need to focus in on data selected on a number of criteria (“dimensions”)
- Logistics Management Institute had tried database analytic services and OLAP tools
- None could handle the volume of data or produce the statistical analysis needed.

Case Study – Office of the Secretary of Defense

Challenges:

- At least 20 different “dimensions” (attributes) needed for user to select subsets of data.
- Percentiles must be calculated on user selected subsets – so pre-aggregation tricks used by many OLAP products won’t work

Case Study – Office of the Secretary of Defense

Before SmartArrays, LMI's best result was a 200 MB Microsoft Pivot table

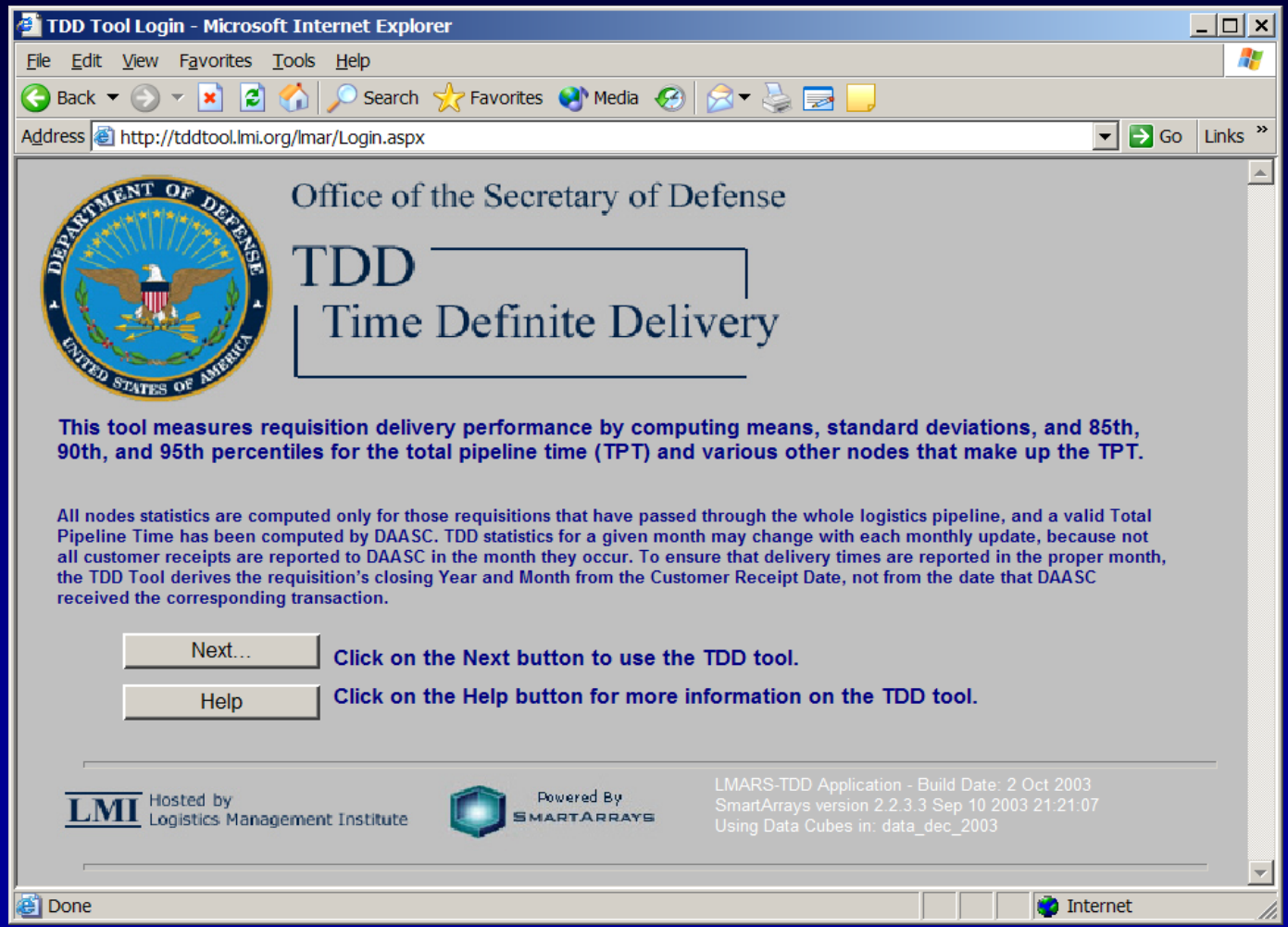
- Couldn't hold more than half the desired dimensions
- A long time to recalculate when the user specified new criteria

Case Study – Office of the Secretary of Defense

SmartArrays Based Solution:

- All the data online
- Handles all the dimensions easily
- Recalculates in as little as 1 second
- User-friendly web site built with ASP.NET

Case Study – Office of the Secretary of Defense



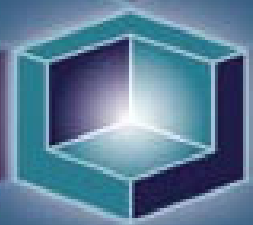
Case Study – Office of the Secretary of Defense

Results:

- Extremely fast calculations
- Data selected and sliced the way the user needs it.
- Same analysis is available to everyone along the supply chain
- Stakeholders can see the same information, spot problems, set goals.

SmartArrays®

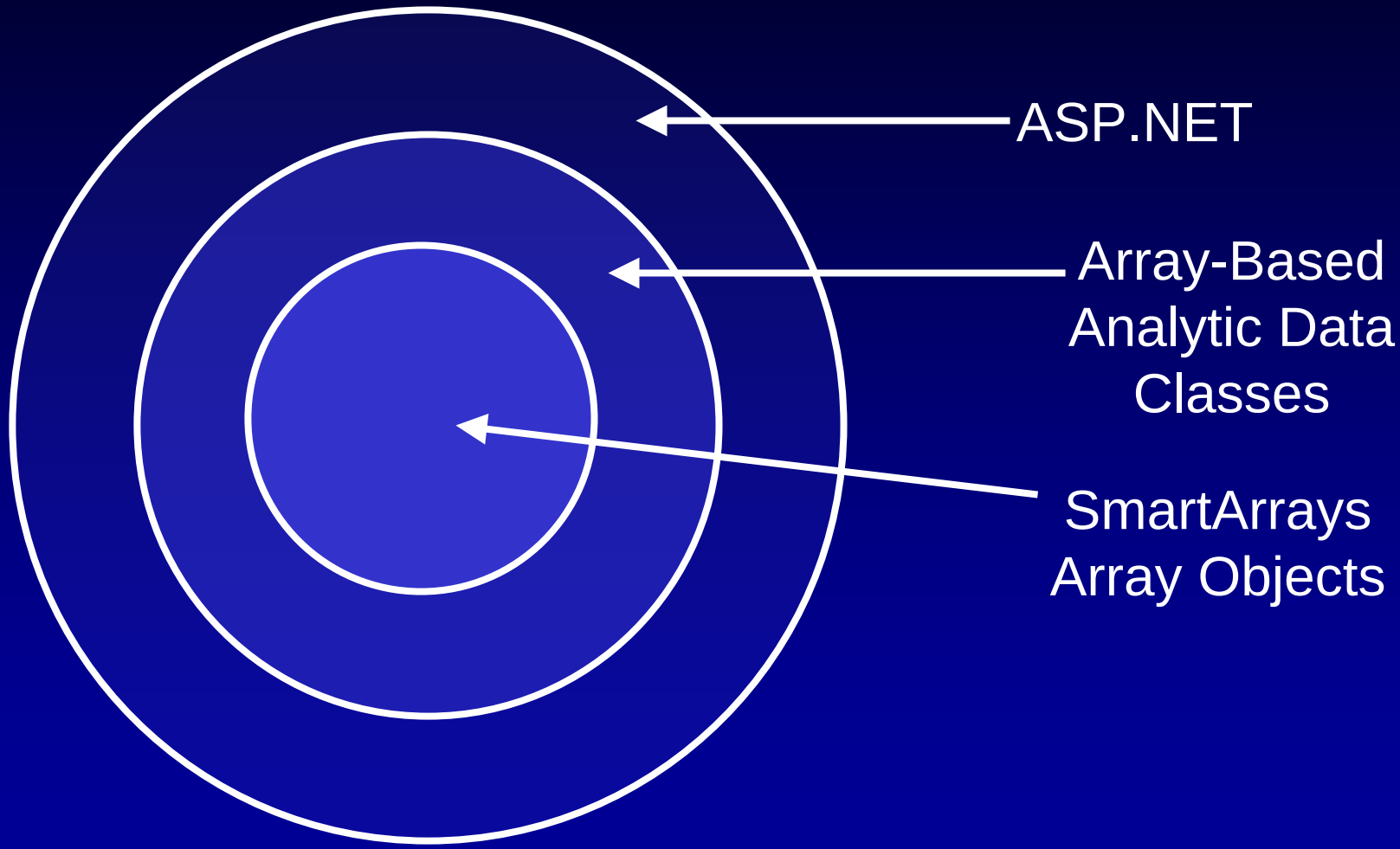
How It Works: Application Architecture



Techniques for Data-Intensive Analytics

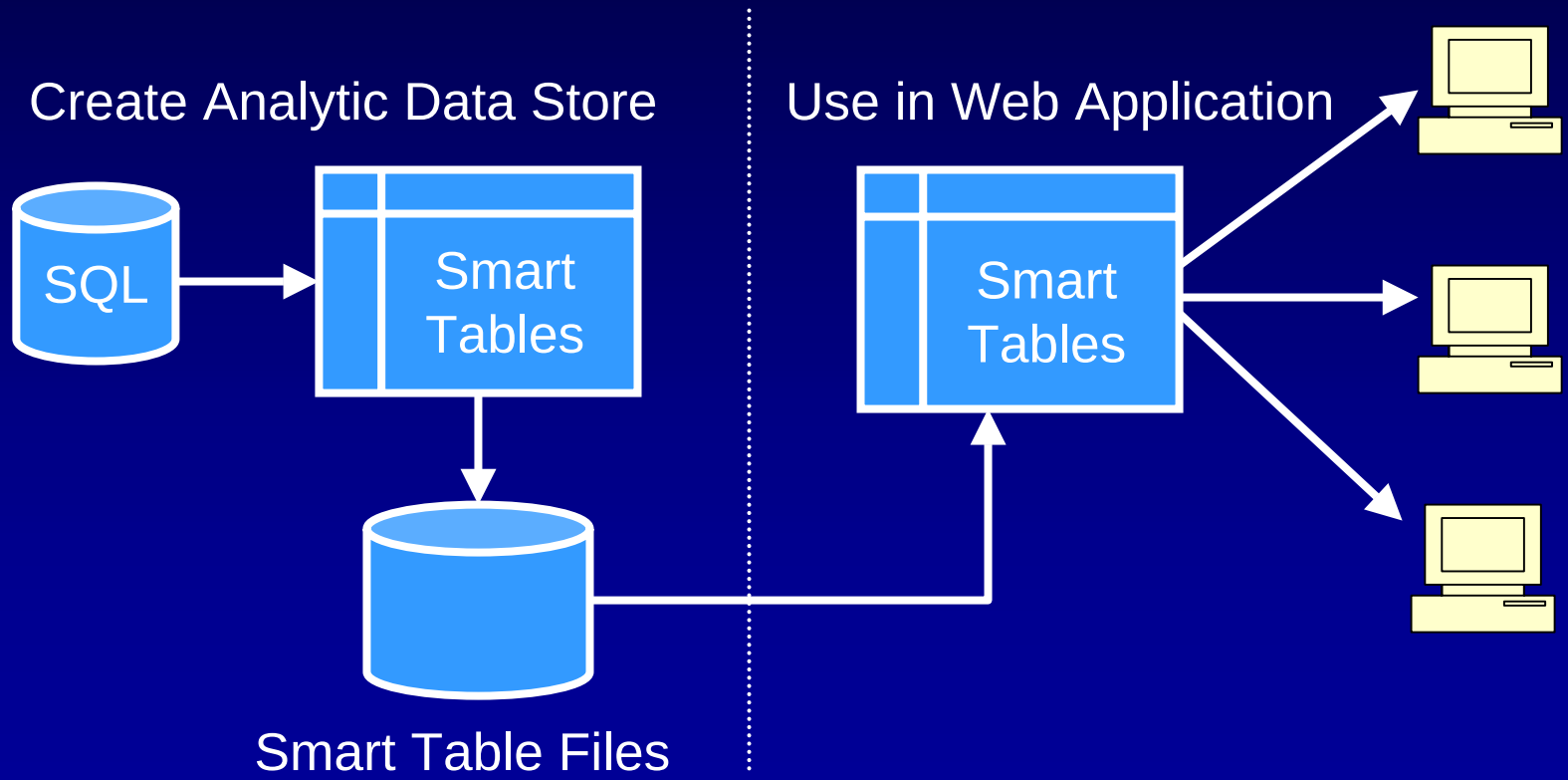
- Extract database data in advance.
- Keep all the data in memory or cached array files
- Use “column-sliced” form of data tables
- Use arrays to store data very compactly
- Bitmap based “slicing” for fast queries
- Highly optimized algorithms based on decades of experience.

TDD Tool Architecture



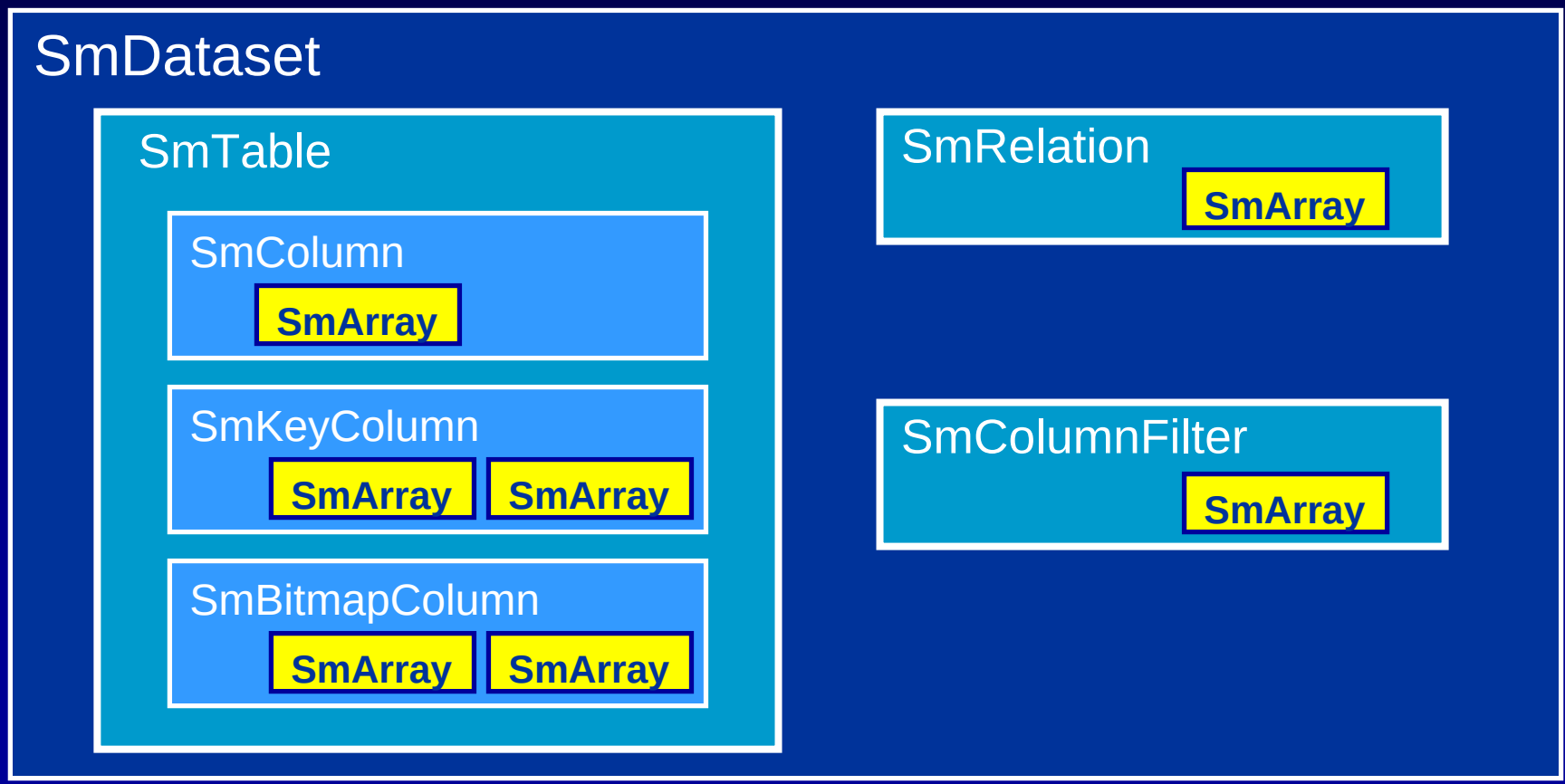
Analytic Data Store

Array-Based “Virtual Database”



Analytic Data Classes

Collection of classes – Java, C++ and C# – that organize data for analysis.



Column Sliced Tables

Conventional Approach

- A database stores data as rows (records)
- DBA adds indexes to improve query speed on columns

SmartArrays Approach

- Smart Tables are column sliced – each field in a vector.
- No indexes needed. Every column can be queried at equal speed.

Column-Sliced Tables

DBMS uses row-sliced table, storing data as records of fields

Product	Quantity	Price
“Apple”	12	4.23
“Orange”	6	1.25
“Banana”	1	0.59

Column-Sliced Tables

- Analytic array-based tables hold all values for each field together in a data vector
- Values Stored as Adjacent Values in Memory

Product	Quantity	Price
“Apple”	12	4.23
“Orange”	6	1.25
“Banana”	1	0.59
…	…	…

Compact Data Format

“Customer” Column in the database:

- 4 million rows
- 100 MB on disk

“Customer” Column in SmartArrays

- 2 MB ***in memory***
- Much, much faster to access

Bitmap Encoded Columns

- Class SmBitmapColumn – works like a column vector of data
- Useful when data has a small number of unique values
- Exploits SmartArrays boolean data format – one bit per value
- Very compact to store
- Incredibly fast to query

SmBitmapColumn

Customer column has 4 million rows but only 4 distinct values: “Air Force”, “Army”, “Navy”, “Marines”

Represent with Arrays:

- Vector of 4 strings: “Air Force”, “Army”, “Navy”, “Marines”

“Air Force”	“Army”	“Navy”	“Marines”

- Vector of 4 bitmaps, each with 4 million bits (500K)

```
0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0…
1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0…
0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0…
0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1…
```

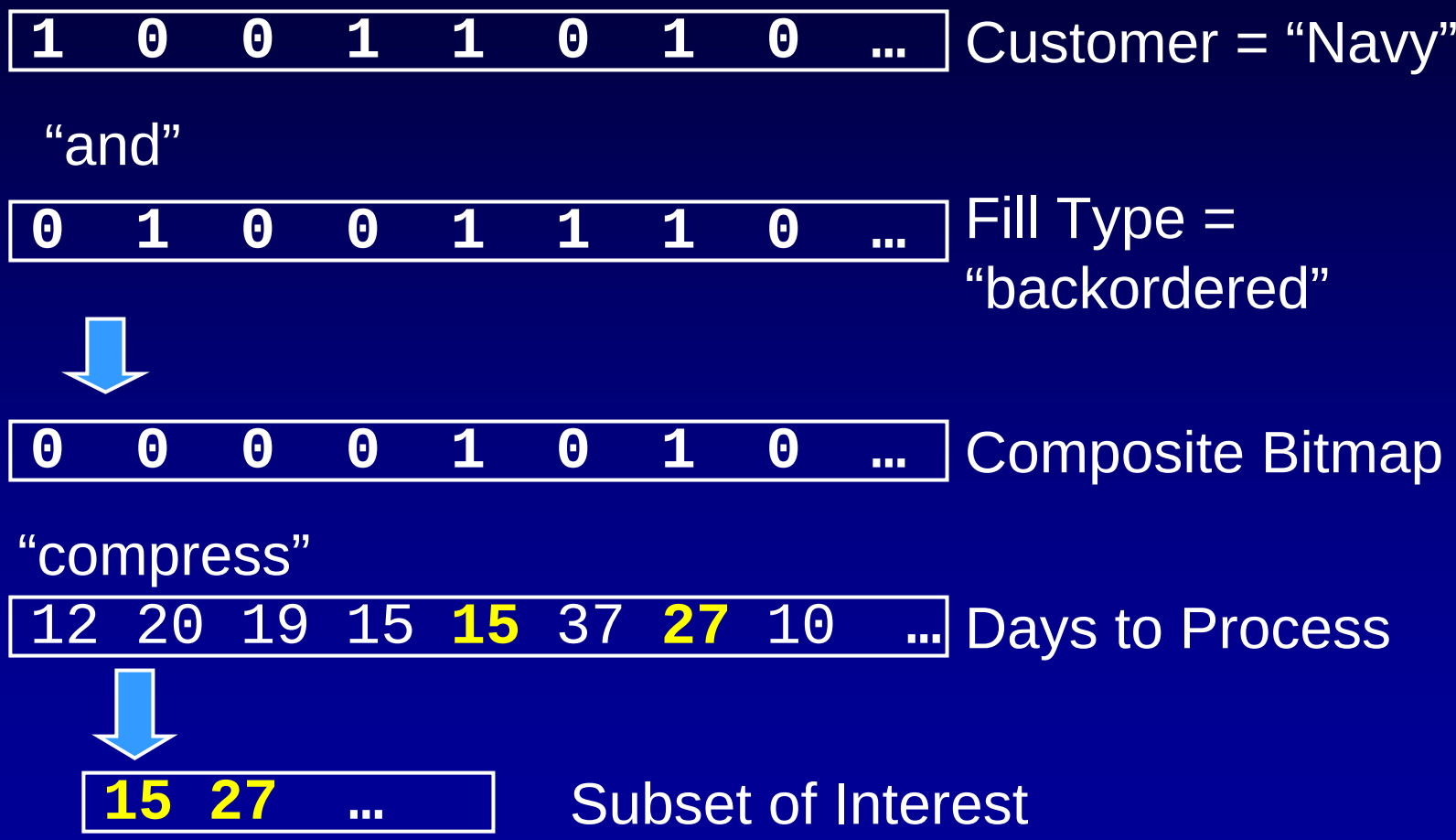
Slice Subsets

- Let the user pick any combination of characteristics to include in analysis.
- Let the user pick any set of dimensions to spread the data over.
- Select data subsets on-the-fly

Slicing with Bitmaps

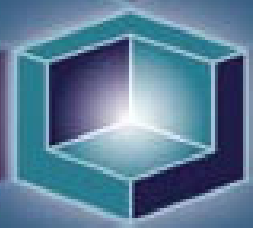
- Select bits for each dimension filter
- Combine with “and” or “or” operations
- Use combined bitmap to “compress” data values of interest
- MUCH faster than SQL queries
- No limit on number of dimensions

Slicing with Bitmaps



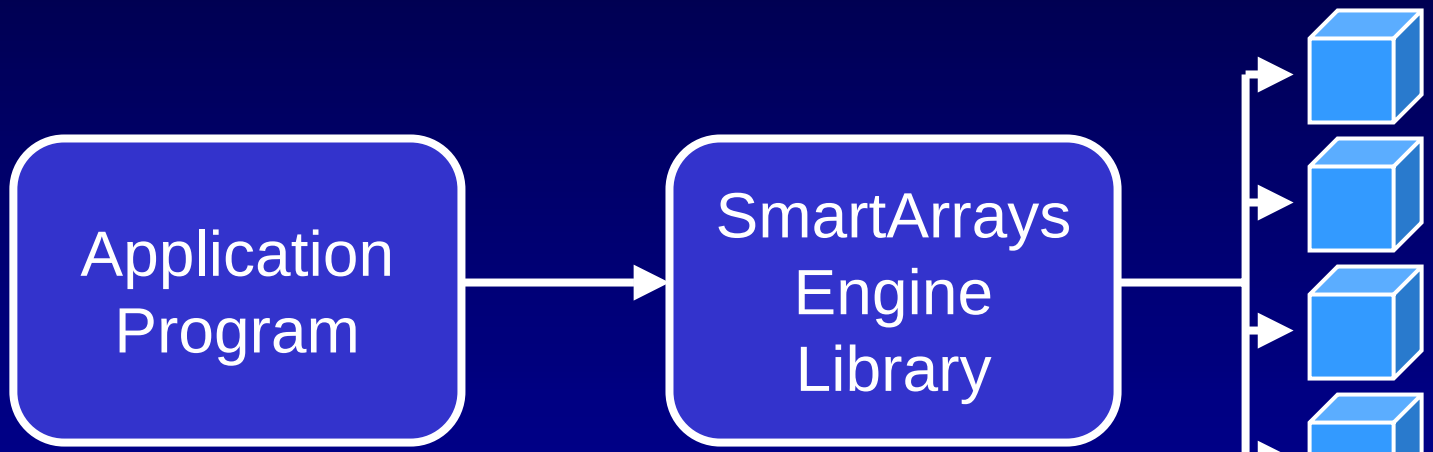
SmartArrays®

Working with SmartArrays



Working With SmartArrays

Physical Architecture:



Application calls the SmartArrays engine

Engine creates arrays and performs array operations

Class SmArray – Arrays as Intelligent Data Objects

- Scalars, vectors, matrices, higher-dimensions
- Datatypes available: bit, char, wchar, string, integer, double, complex, nested, mixed

Developing with SmartArrays

- One class – SmArray – for all arrays, any type, any shape, any structure
- More than 200 built-in bulk data array methods
- Each method returns a new SmArray
- Methods to describe arrays and exchange data with other classes
- Array-oriented database adapter

Creating an Array

Make scalars:

```
SmArray s = SmArray.scalar(“Ford”); // a string
SmArray n = SmArray.scalar( 1.24 ); // number
```

Make Vectors:

```
SmArray sv =
   SmArray.vector(“Ford”,”Daimler”,”GM”);
SmArray nv = SmArray.vector(10,20,30);
SmArray seq = SmArray.sequence(5,100);
seq.show();
    100 101 102 103 104 [debug output]
```

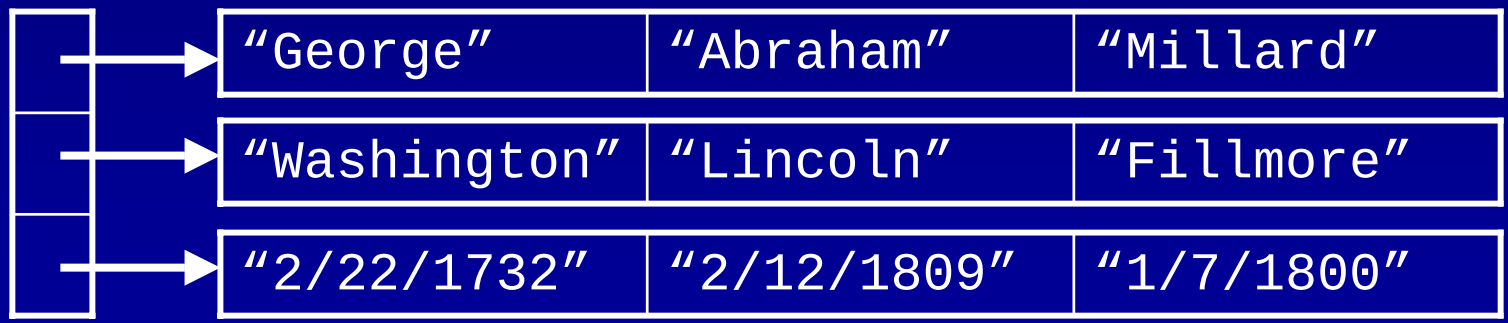
Filling Array from Database

Create and Connect a Database Adapter

```
SmDatabase db = new SmDatabase();
db.connect( connection_string );
```

SQL Query produces vector of column vectors:

```
SmArray columns = db.queryVector (
  “SELECT first_name,last_name,dob from
  employee” );
```



Working with Arrays

Each method returns an array as a result

```
  v1.show();
 10 20 30
  v2.show();
 100 200 300
  v1.plus(v2).show();
  110 220 330
```

Calculate on entire array at once.

```
v1 = v1.times(0.1234);
```

Type converts to floating-point automatically

Combining Arrays

```
m1 = m1.catenate(m2);
```

1	2	3
4	5	6
7	8	9

10	20
10	20
10	20

1	2	3	10	20
4	5	6	10	20
7	8	9	10	20

```
m1 = m1.rowCat( vector )
```

1	2	3	10	20
4	5	6	10	20
7	8	9	10	20

30	40	50	60	70

1	2	3	10	20
4	5	6	10	20
7	8	9	10	20
30	40	50	60	70

Selecting From Arrays

m1

1	2	3	10	20
4	5	6	10	20
7	8	9	10	20
30	40	50	60	70

Selecting rows or columns:

```
SmArray v = m1.row(2);
```

7	8	9	10	20

Selecting with subscripts:

```
v.index( SmArray.vector(4,1));
```

20	8

Selecting with Bitmaps

Vector of 10,000,000 numbers

```
12 33 192 43 291 478 12 182 8 …
```

Find all the values between 100 and 200:

```
bits = v.ge(100).and( v.lt(200) );
```

```
0 0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 …
```

0.38 secs

Select the subset:

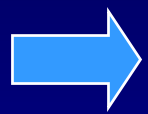
```
v2 = bits.compress( v );
```

```
192 182 118 177 …
```

0.04 secs if 50,000 values in result

Cross Tabulation

Product	State	Quantity
Beer	VA	10
Wine	VA	20
Beer	MD	20
Wine	MD	10
Wine	MD	40
Cheese	MD	20
Milk	VA	30
Wine	DC	20
Milk	DC	20
Milk	VA	40
Cheese	DC	5
Beer	MD	20
Beer	VA	10
Cheese	MD	20
…	…	…



Simple cross-tabulation of totals by two dimensions.

	VA	MD	DC
Beer	20	40	0
Wine	50	50	20
Milk	70	0	20
Cheese	0	40	5

Cross Tabulation

Perform the database query, and create three column vectors from the result:

```
SmArray table = db.queryVector(
   “SELECT product,state,quantity FROM sales” );
SmArray product = table.pickBy(0);
SmArray state = table.pickBy(1);
SmArray quantity = table.pickBy(2);
```

Cross Tabulation

Get the unique product and state values:

```
SmArray product_keys = product.unique();
```

Beer	Wine	Milk	Cheese

```
SmArray state_keys = state.unique();
```

MD	VA	DC

Cross Tabulation

Map each row to the set of unique keys:

```
SmArray row_inds = product_keys.lookup(product);
```

0	1	0	1	1	2	3	1	2	…

```
SmArray col_inds = state_keys.lookup(state);
```

0	0	1	1	1	1	2	2	0	…

Cross Tabulation

Glue the row and column inds together:

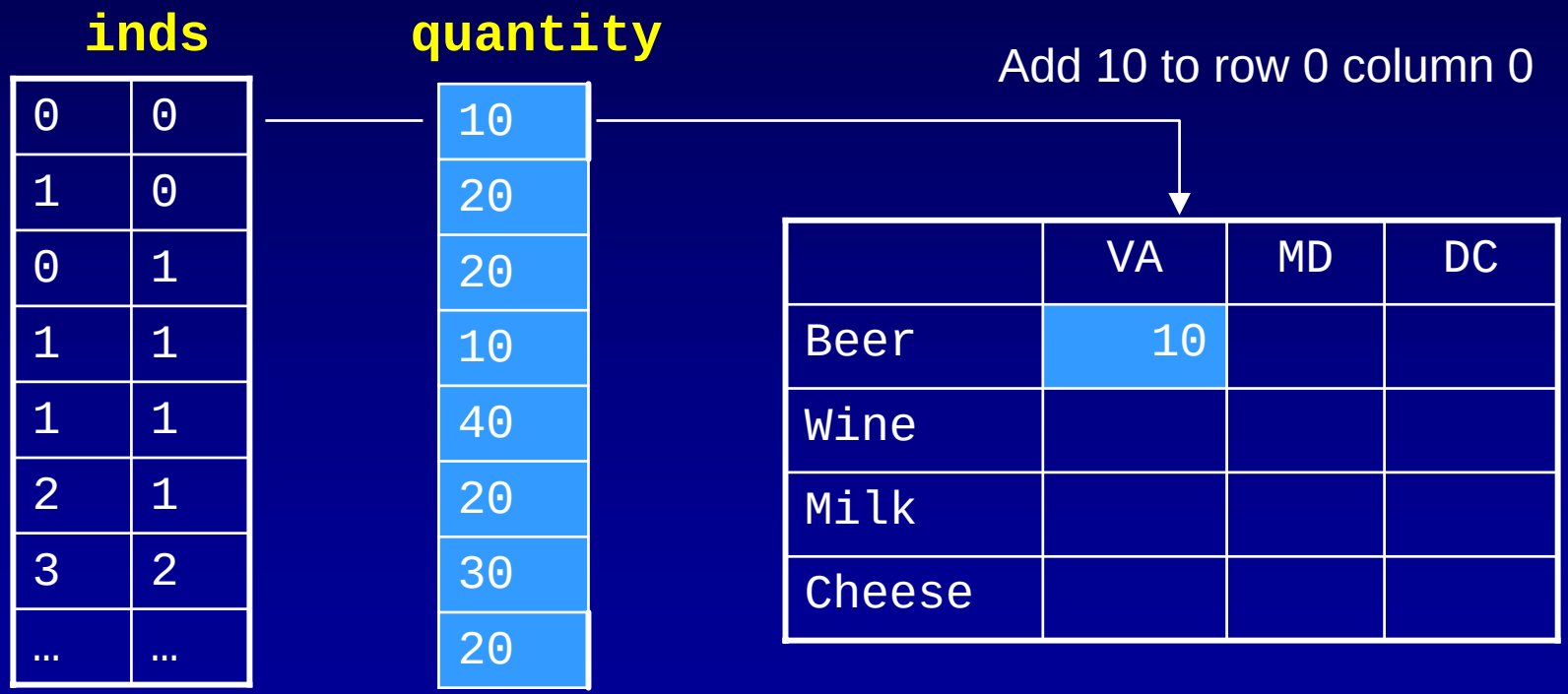
```
SmArray inds = row_inds.colCat(col_inds);
```

inds

0	0
1	0
0	1
1	1
1	1
2	1
3	2
…	…

Cross Tabulation

Total quantity values into result matrix:



Cross Tabulation

Create the matrix of zeroes:

```
SmArray xtab = SmArray.scalar(0).reshapeBy(
    product_keys.getCount(), state_keys.getCount() );
```

0	0	0
0	0	0
0	0	0
0	0	0

Perform the cross-tab aggregation with one function:

```
xtab.selectUpdateBySubscript(
     Sm.plus, quantity, inds );
```

20	40	0
50	50	20
70	0	20
0	40	5

Cross Tabulation Function

```
SmArray crosstab( SmArray rvals, SmArray cvals, SmArray datavals )
{
    // rvals, cvals: values for rows and columns of cross tabs table
    // datavals:  numeric values to aggregate
    SmArray row_keys = rvals.unique();
    SmArray col_keys = cvals.unique();
    SmArray xtab = SmArray.scalar(0).reshapeBy(
       row_keys.getCount(), col_keys.getCount() );
    xtab.selectUpdateBySubscript( Sm.plus, datavals,
       row_keys.lookup(rvals).colCat( col_keys.lookup(cvals)));
    return xtab;
}
```

Works with any type of data for keys

Cross tabulates a 100,000 row data set in < 0.2 secs.

A Full Range of Functions

Arithmetic, Comparison, Trigonometric

Mathematical: statistics, radix arithmetic, matrix inverse, etc.

Selection, filtering, subscripting

Sorting and Searching

Date and Time

Database and File I/O,

HTML, XML, formatting, string processing

More than 250 in all.

Summary

With SmartArrays, data-intensive analytic applications are simple to build and very fast to execute.

It’s Easy to Get Started with SmartArrays

- Evaluation editions available for all products.
- See the value immediately with a pilot project
- Use SmartArrays to build “special case” solutions very rapidly
- Train your own development team, or engage a SmartArrays consulting partner
- SmartArrays staff available to mentor your team on array technology
- Licenses: developer, server, and runtime available

Thanks for Listening

For more information:
http://www.smartarrays.com
Email: sales@smartarrays.com
Phone: +1 703 326 0066