SciQL
A Query Language for Science Applications

M. Kersten, Y. Zhang, M. Ivanova, N. Nes
CWI Amsterdam

Array Database Workshop
March 25th, 2011
Who needs arrays anyway?

Seismology – 1-D time-series, 3-D spatial data
Astronomy – temporal ordered rasters
Climate simulation – temporal ordered grid
Remote sensing – images of 2-D or higher
Genomics – ordered DNA strings

Scientists love arrays:

- HDF5, NETCDF, FITS, MSEED, ...

but also use:

- lists, tables, XML, ...
Research issues already in the 80’s

SQL language extension (add notion of order):
  - RasQL, AQuery, SRQL, ...

SQL:1999, SQL:2003
  - collection type, C-style arrays

Algebraic frameworks
  - (S)RAM, AQL, AML, ...
Arrays In DBMS

- DBMS support
  - OODB, multi-dimensional DBMS, Sequence DBMS, ...
  - the Longhorn Array Database

- RasDaMan
  - Array in chunks as BLOB
  - Array query optimisation on top of DBMS
  - Known to work up to 12 TBs!

- PostgreSQL 8.1

- SciDB
  - Array DBMS from scratch
  - Overlapping chunks for parallel execution
What is the problem with RDBMS?

- Appropriate array denotations?
- Functional complete operation set?
- Size limitations due to (BLOB) representations?
- Existing foreign files?
- Scale?
- ...
An extension of SQL:2003 (pronounced as ‘cycle’)

- Array as *first class citizens* of DBMS
- Seamless integration of tables and arrays
- Named dimensions with constraints
- Flexible structure-based grouping

Seismology use case
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);
Array Definitions

Fixed array

```
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);
```

Unbounded array

```
CREATE ARRAY A2 (  
x INT DIMENSION,  
y INT DIMENSION,  
v FLOAT DEFAULT 0.0  
);
```
**Fixed array**

CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);  

```
 0.0 0.0 0.0 0.0  
 0.0 0.0 0.0 0.0  
 0.0 0.0 0.0 0.0  
 0.0 0.0 0.0 0.0  
 0.0 0.0 0.0 0.0  
```

**Unbounded array**

CREATE ARRAY A2 (  
  x INT DIMENSION,  
  y INT DIMENSION,  
  v FLOAT DEFAULT 0.0  
);  

```
0.0 4.5 0.0 0.0  
5.5 0.0 0.0 0.0  
```

INSERT INTO A2 VALUES  
(1,0,5.5), (1,1,0.4), (2,2,4.5);
Array Definitions

**Fixed array**

CREATE ARRAY A1 (
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Unbounded array**

CREATE ARRAY A2 (
  x INT DIMENSION,
  y INT DIMENSION,
  v FLOAT DEFAULT 0.0
);

INSERT INTO A2 VALUES
  (1,0,5.5), (1,1,0.4), (2,2,4.5);

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Array Dimensions

- Fixed dimensions: \([\text{start}:\text{final}:\text{step}]\)
- INT dimension: \([\text{size}]\)
- Unbounded dimensions: \([(\text{start}|\ast) : (\text{final}|\ast) : (\text{step}|\ast)]\)
- Dimension data type: scalar data types
- Time series:

```sql
CREATE ARRAY A1 (  x INT DIMENSION[0:4:1],  y INT DIMENSION[0:4:1],  v FLOAT DEFAULT 0.0 );
```

```sql
CREATE ARRAY A2 (  x INT DIMENSION,  y INT DIMENSION,  v FLOAT DEFAULT 0.0 );
```

```sql
CREATE ARRAY Experiment (  time TIMESTAMP DIMENSION [TIMESTAMP '2011-03-25': * : INTERVAL '1' MINUTE],  data FLOAT );
```
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);  

CREATE TABLE T1 (  
x INT,  
y INT, PRIMARY KEY (x,y),  
v FLOAT DEFAULT 0.0  
);
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  );

SELECT * FROM A1;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>

CREATE TABLE T1 (  
x INT,  
y INT, PRIMARY KEY (x,y),  
v FLOAT DEFAULT 0.0  );

SELECT * FROM T1;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Array versus Table

- **Array A1**
  - A collection of a priori defined tuples
  - To be updated with INSERT/DELETE (and UPDATE)
  - Indexed by dimension expressions
  - Default value for non-dimensional attributes (i.e., cells)

- **Table T1**
  - A collection of tuples
  - Explicitly create/remove with INSERT/DELETE
  - Indexed by a (primary) key
  - Default value for each column

CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);

CREATE TABLE T1 (  
  x INT,
  y INT, PRIMARY KEY (x,y),
  v FLOAT DEFAULT 0.0
);
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);

SELECT x, y, v FROM A1;

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0</td>
</tr>
</tbody>
</table>
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);  

SELECT x, y, v FROM A1;

```
x y v
0 0 0.0
0 1 0.0
0 2 0.0
0 3 0.0
1 0 0.0
1 1 0.0
1 2 0.0
1 3 0.0
2 0 0.0
2 1 0.0
2 2 0.0
2 3 0.0
3 0 0.0
3 1 0.0
3 2 0.0
3 3 0.0
```
CREATE TABLE T2 (  
x INT, y INT, v FLOAT 
);

INSERT INTO T2 VALUES  
(1,0,5.5), (1,1,0.4),  
(2,2,4.5), (1,1,1.3);

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.3</td>
</tr>
</tbody>
</table>
CREATE **TABLE T2** (  
  x INT, y INT, v FLOAT  
);  

INSERT INTO **T2** VALUES  
(1,0,5.5), (1,1,0.4),  
(2,2,4.5), (1,1,1.3);  

SELECT [x], [y], v FROM T2;  

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.3</td>
</tr>
</tbody>
</table>

An unbounded array  
- min/max of dimensions are derived from the minimal bounding rectangle  
- non-dimensional attributes inherit default column values  
- duplicates are overwritten
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  );

DELETE FROM A1 WHERE x = 1;

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>null</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>null</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>null</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>null</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);  

DELETE FROM A1 WHERE x = 1;  

creates holes in the array
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0 );

INSERT INTO A1 VALUES (1,1,0.5), (2,1,0.5), (3,1,0.5);

```sql
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0 );

INSERT INTO A1 VALUES (1,1,0.5), (2,1,0.5), (3,1,0.5);
```
CREATE ARRAY A1 ( 
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);

INSERT INTO A1 VALUES (1,1,0.5), (2,1,0.5), (3,1,0.5);

set (change) values of cells
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT -1.0  
);  

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1.0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td></td>
</tr>
</tbody>
</table>
CREATE ARRAY A1 (  
x INT DIMENSION [0:4:1],  
y INT DIMENSION [0:4:1],  
v FLOAT DEFAULT -1.0 )
);

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

CREATE ARRAY VIEW A2 (  
x INT DIMENSION [-1:5:1],  
y INT DIMENSION [-1:5:1],  
w FLOAT DEFAULT 0.0  
) AS  
SELECT x-1, y, v FROM A1 WHERE x > 1  
UNION  
SELECT x, y, 1.0 FROM A1 WHERE x = 3;

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```

```
CREATE ARRAY A1 (x INT DIMENSION[0:4:1], y INT DIMENSION[0:4:1], v FLOAT DEFAULT -1.0);

INSERT INTO A1 VALUES (1,1,0.5), (2,1,0.5), (3,1,0.5);

CREATE ARRAY VIEW A2 (x INT DIMENSION [-1:5:1], y INT DIMENSION [-1:5:1], w FLOAT DEFAULT 0.0)
  AS
  SELECT x-1, y, v FROM A1 WHERE x > 1
  UNION
  SELECT x, y, 1.0 FROM A1 WHERE x = 3;

SELECT x, y, 1.0 FROM A1 WHERE x = 3;
CREATE ARRAY A1 (  
  x INT DIMENSION [0:4:1],  
  y INT DIMENSION [0:4:1],  
  v FLOAT DEFAULT -1.0  
);  

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

CREATE ARRAY VIEW A2 (  
  x INT DIMENSION [-1:5:1],  
  y INT DIMENSION [-1:5:1],  
  w FLOAT DEFAULT 0.0  
) AS  
SELECT x-1, y, v FROM A1 WHERE x > 1  
UNION  
SELECT x, y, 1.0 FROM A1 WHERE x = 3;
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT -1.0  
);  

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

CREATE ARRAY VIEW A2 (  
x INT DIMENSION [-1:5:1],  
y INT DIMENSION [-1:5:1],  
w FLOAT DEFAULT 0.0  
) AS  
SELECT x-1, y, v FROM A1 WHERE x > 1  
UNION  
SELECT x, y, 1.0 FROM A1 WHERE x = 3;
CREATE ARRAY A1 (  
x INT DIMENSION [0:4:1],  
y INT DIMENSION [0:4:1],  
v FLOAT DEFAULT -1.0  
);  

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

CREATE ARRAY VIEW A2 (  
x INT DIMENSION [-1:5:1],  
y INT DIMENSION [-1:5:1],  
w FLOAT DEFAULT 0.0  
) AS  
SELECT x-1, y, v FROM A1 WHERE x > 1  
UNION  
SELECT x, y, 1.0 FROM A1 WHERE x = 3;  

SELECT x, y, 1.0 FROM A1 WHERE x = 3;
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT -1.0  );

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

CREATE ARRAY VIEW A2 (  
x INT DIMENSION [-1:5:1],  
y INT DIMENSION [-1:5:1],  
w FLOAT DEFAULT 0.0  ) AS  
SELECT x-1, y, v FROM A1 WHERE x > 1  
UNION  
SELECT x, y, 1.0 FROM A1 WHERE x = 3;
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0 );
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);

INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];  

y
null
null
0.0 0.0 0.0 0.0

null
0.0 0.0 0.0 0.0

null
0.0 0.5 0.5 0.5

null
0.0 0.0 0.0 0.0

Anchor point
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
  y INT DIMENSION[0:4:1],  
  v FLOAT DEFAULT 0.0  
);
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  );  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 ( 
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);
INSERT INTO A1 VALUES
  (1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1
GROUP BY A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
  x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0
);  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY A1[x:x+2][y:y+2];

Array Tiling

tiling ≠ windowing
CREATE ARRAY A1 (  
x INT DIMENSION [0:4:1],  
y INT DIMENSION [0:4:1],  
v FLOAT DEFAULT 0.0 )  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);
INSERT INTO A1 VALUES
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1
GROUP BY DISTINCT A1[x:x+2][y:y+2];

<table>
<thead>
<tr>
<th>y</th>
<th>null</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>null</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Anchor point
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  );
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  x INT DIMENSION[0:4:1],  y INT DIMENSION[0:4:1],  v FLOAT DEFAULT 0.0 );
INSERT INTO A1 VALUES  (1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1  GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 ( 
   x INT DIMENSION[0:4:1],
   y INT DIMENSION[0:4:1],
   v FLOAT DEFAULT 0.0
);
INSERT INTO A1 VALUES
   (1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 ( 
  x INT DIMENSION[0:4:1],
  y INT DIMENSION[0:4:1],
  v FLOAT DEFAULT 0.0
);
INSERT INTO A1 VALUES 
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1 
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  
);  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1  
GROUP BY DISTINCT A1[x:x+2][y:y+2];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0  );  
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);  

SELECT [x], [y], AVG(v) FROM A1[1:*][1:*]  
GROUP BY DISTINCT A1[x-1][y], A1[x][y-1],  
A1[x][y], A1[x+1][y], A1[x][y+1];
CREATE ARRAY A1 (  
x INT DIMENSION[0:4:1],  
y INT DIMENSION[0:4:1],  
v FLOAT DEFAULT 0.0 );
INSERT INTO A1 VALUES  
(1,1,0.5), (2,1,0.5), (3,1,0.5);

SELECT [x], [y], AVG(v) FROM A1[1:*][1:*]  
GROUP BY DISTINCT A1[x-1][y], A1[x][y-1],  
A1[x][y], A1[x+1][y], A1[x][y+1];
Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long
Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

```
CREATE TABLE MSeed (
  station VARCHAR(10);
  ts ARRAY (  
    tick TIMESTAMP DIMENSION  
      [* : * : INTERVAL '0.01' SECOND],  
    data DECIMAL(8,6)  
  )  
);  
```
Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec/15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

--- avg of 2 sec. windows:

```sql
SELECT A.station, A.ts.tick, AVG(A.ts.data)
FROM MSeed AS A
GROUP BY
    A.ts[tick - INTERVAL '2' SECOND : tick];
```
Seismology Use Case

Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

CREATE TABLE Event(
    station STRING,
    tick TIMESTAMP,
    ratio FLOAT)
AS
SELECT A.station, A.ts.tick,
    AVG(A.ts.data)/AVG(B.ts.data) AS ratio
FROM MSeed AS A, MSeed AS B
WHERE A.station = B.station
    AND A.ts.tick = B.ts.tick
GROUP BY
    A.ts[tick - INTERVAL '2' SECOND : tick],
    B.ts[tick - INTERVAL '15' SECOND : tick]
HAVING AVG(A.ts.data)/AVG(B.ts.data) > ?delta
WITH DATA;
Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

CREATE TABLE Neighbors(
  head STRING,
  tail STRING,
  delay TIMESTAMP,
  weight FLOAT
);

-- detect isolated errors by direct environment
-- using wave propagation statics
Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

-- detect false positives:

```
SELECT A.station, A.tick
FROM Event AS A, Event AS B, Neighbor AS N
WHERE A.station = N.head
  AND B.station = N.tail
  AND B.tick = A.tick + N.delay
  AND A.ratio > B.ratio * N.weight;
```

-- remove the false positives from Event
Seismology Use Case

Recent aftershock in Chili

- 2TB waveform data at 100Hz
- detecting seismic events using STA/LTA (e.g., 2 sec / 15 sec)
- remove false positives
- window-based 3 min. cuts
- heuristic tests

Current problems

- accessing waveform files too slow
- unpacking and positioning MSEED data takes too long

```sql
-- pass time series to a UDF, written in, e.g., C:
SELECT A.station, myfunction(A.ts)
FROM MSeed A, Event B
WHERE A.station = B.station
  AND A.ts.tick = B.tick
GROUP BY DISTINCT
  A.ts[tick - INTERVAL '3' MINUTE : tick];
```
Conclusion

- Appropriate array denotations
- Functional complete operation set
- Size limitations due to (blob) representations
- Existing foreign files?
- Scale?

- An Array DBMS for sciences
  - Symbiosis of relational and array paradigms